

FM TOWNS

FUJITSU FM SERIES PERSONAL COMPUTER

F-BASIC386 V2.1リファレンス

（F-BASIC386V2.1, F-BASIC386V2.1コンパイラ対応）

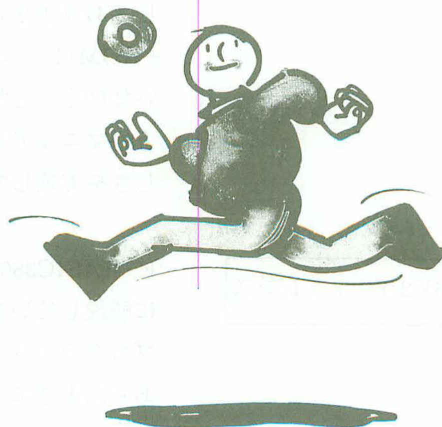
FUJITSU

本マニュアルには，“外国為替及び外国貿易管理法”に基づく特定技術が含まれています。したがって，本マニュアルまたはその一部を輸出する場合には，同法に基づく許可が必要とされます。
富士通株式会社

FM TOWNS

FUJITSU FM SERIES PERSONAL COMPUTER

F-BASIC386 V2.1リファレンス



ごあいさつ

このたびは，F－BASIC386V2．1をお買い求めいただき，誠にありがとうございます。
FMTOWNSは，ひとりひとりが快適な生活を創り出すことを応援するパソコンです。
F－BASIC386は，このFMTOWNSの機能を活用してプログラムを作るためのやさしいプログラミング言語です。
本書は，F－BASIC386のプログラミング上の規約と全命令の動作を解説しています。
本書が皆様のお役に立つことを願っております。

1992年11月

本書をお読みになる前に

■FMTOWNSの基本的な操作については、FMTOWNS本体に添付のマニュアルで説明しています。本書をお読みになる前に必ずご覧ください。

■F−BASIC386に添付のマニュアルについて

F−BASIC386には次の３冊のマニュアルが付いています。

F−BASIC386V2.1ガイド	このマニュアルでは、起動と終了、F−BASIC386コンパイラの操作方法、エディタの操作方法などを説明しています。また、盛りだくさんのサンプルプログラムも用意してあります。サンプルプログラムはCD−ROMに入っていますのですぐに使っていただくことができます。 このマニュアルを読んでプログラムする楽しさを実感してください。
F−BASIC386V2.1リファレンス	F−BASIC386の全ての機能について詳細に解説しています。 プログラミング中にわからない命令などがあったときにこのマニュアルをお読みください。
F−BASIC386V2.1ポケットブック	F−BASIC386の全命令をコンパクトなポケットブックにまとめました。形式と機能だけの簡単な一覧ですが、必ずお役に立つと思います。プログラミングのときは、ぜひそばに置いてご活用ください。

■F−BASIC386には、上記の３冊のマニュアルの他に、CD−ROMの中にオンラインチュートリアルマニュアル(ベーシックアイランドの冒険)が入っています。F−BASIC386の入門にお役立てください。

本書の読み方

本書の構成は次のようになっています。

第1章　プログラミング上の規約	BASICの行、変数、定数、演算などについての基本的な概念や規約を説明しています。
第2章　命令の概要と使い方	命令を機能別に分類して、命令の基本的な使い方と相互関係について説明しています。 ある機能を実現するのに、どの命令を使用すればよいかを知ることができます。
第3章　BASICの命令	F−BASIC386の全命令をアルファベット順に説明していますので、命令のつづりをキーに、命令の形式や機能説明を探し出すことができます。
第4章　機械語プログラム	機械語プログラムの呼び出し、実行について説明しています。
付録	エラーメッセージ一覧、音色番号一覧、F−BASIC命令互換一覧、キャラクタコード、ESCシーケンス、インタプリタとコンパイラの機能の違い、数学関数の各リストを載せています。

知りたい情報を探し出すのに、次のような方法があります。

・使用目的から命令のつづりを探す………………第2章または索引
・命令のつづりから命令の形式や説明を探す……第3章
・用語説明または解説を探す………………………索引

表記について

本書ではF−BASIC386をBASICと表記しています。


−プログラミングまたはオペレーション上の注意事項が書いてあります。


−用語の意味を説明しています。


−詳細説明または関連説明のある参照箇所を指しています。

インタプリタ −インタプリタでのみ有効な機能であることを示しています。

コンパイラ −コンパイラでのみ有効な機能であることを示しています。

目　次

第1章　プログラミング上の規約

第 2 章　命令の概要と使い方

第 3 章　BASICの命令

第 4 章　機械語プログラム

付　　録

索　　引

第1章

プログラミング上の規約

BASICが扱う命令や式、ファイルや画面についての規約を説明します。

1. 行

プログラムは、行番号がついている命令(行)の集まりです。

行を構成する要素とその記述方法について説明します。

行は、必ず行番号で始まり、ラベル名、文、コメントの各要素のいずれか1つまたは全部から構成されます。

1行の長さは行番号を含め、255文字以下でなければなりません。

行番号　ラベル名　文：文…　'コメント

```
例： 1000   *SUB  CIRCLE(300,200),200,,,,F   'GRAPH
     └行番号 └ラベル名          └文          └コメント
```



プログラムを見やすくするため、自由に1文字以上の空白を挿入できます。

ただし、次に示す場所には空白を挿入できません。

- 行番号の途中
- 予約語の途中(ただしGOとTOの間、GOとSUBの間には空白を入れてもかまいません)
- 関数名、変数名、数値定数の途中



- 予約語の前後は、1文字以上の空白または特殊記号で切り離す必要があります。

行を入力し、リターンキーを押すか、またはカーソル移動キーによってカーソルが入力中の行を離れると、その行がプログラムメモリに記憶されます。

● 行番号

プログラムがメモリに記憶されるときの行の順序およびプログラムが実行されるときの順序を示します。

1つのプログラムに同じ行番号の行が2つ以上存在することはできません。

プログラムは、行番号の小さいものから順に実行されます。

行番号は1～65529の範囲の整数です。

● ラベル名

ラベルは、行に対してつけた名前です。行番号が各行に必ずつけなければならないのに対し、ラベルはつけなくてもかまいません。

ラベル名は、その行の位置をわかりやすくします。また、プログラムを分岐させる命令(GOTO, GOSUB)の、分岐先指定に使います。

プログラム内に同じラベル名があるときは、常に最初のラベル名が参照され、エラーとはなりませんが、予期しない動作をすることがあります。

ラベル名は、必ず行番号の直後に記述します。

```
                                          ┌(空白)┐
行番号  ＊日本語文字およびANK文字の列 ┤       ├ 文：文…
        └──ラベル名                    └ ：   ┘
```

＊：ラベル名は、必ずANK文字のアスタリスクで始まります。

日本語文字およびANK文字の列：

ラベル名の先頭に、ANK文字の数字を使うことはできません。

ANK文字の英字の大文字と小文字は、区別されません。

日本語文字とANK文字の混在も許されています。

ラベル名の長さは、そのラベル名を含む1行の長さが255文字以内であれば何文字であってもかまいません。

行番号とラベル名だけの行も作成できます。

ラベル名に続けて文を記述するときは、コロン(:)または空白( )により区切らなければなりません。

● 文

文は命令とオペランドによって構成されます。

例：CIRCLE　(300,200),200,,,,,F
　　└命令　　└オペランド

●実行命令とオペランドにより構成された文を実行文といいます。

——実行命令は、BASICが実行すべきことを命令し、実動作を伴います。

●非実行命令とオペランドにより構成された文を非実行文といいます。

——非実行命令は、他の命令によって参照されたり、実行の際の数値の設定などを行います。

●1つの行に、2つ以上の文を記述することができます。

——各文の間は、コロン(:)で区切ります。

このように複数個の文を記述した行を、マルチステートメントといいます。

● コメント

行の終わりにはコメントを記述できます。

コメントは、直前の文と半角のシングルクォーテーション（'）により分離します。

また、行番号とコメントのみの行も作成できます。

例：1200　'DATA INPUT OK/NG ?
　　└行番号　└コメント

コメントはREM命令でも記述できます。REMの後にコメントを記述します。

例：1200 REM 'DATA INPUT OK/NG ?
　　└行番号　└コメント

2. 文字セット

BASICは次の文字セットを使用できます。

文字の種類	説　　　明
ANK文字の英字	A～Zの26文字で、それぞれ大文字と小文字があります。
ANK文字の数字	0から9までの10文字です。
ANK文字のカナ文字	アイウエオ～ン ﾞ ﾟ
ANK文字の特殊記号	特別な意味をもつ文字として使われます。
日本語文字	JIS第1水準2,965文字、JIS第2水準3,384文字，非漢字453文字および外字からなります。非漢字とは、ひらがな、カタカナ、英数字、記号などの文字です。

ANK文字は1文字1バイト、日本語文字は1文字2バイトでメモリに格納されます。

画面上でANK文字は半角、日本語文字は全角で表示されます。

特殊記号一覧表

文字	名　　称	文字	名　　称
	ブランクまたはスペース	!	エクスクラメーションマーク または感嘆符
=	等価記号または代入記号	&	アンパサント
+	正符号または加算記号	@	アットマーク
−	負記号，演算記号 または ハイフン	,	コンマ
*	アスタリスクまたは乗算記号	.	ピリオドまたは小数点
/	スラッシュまたは除算記号	'	シングルクォーテーション
¥	円記号または整数除算記号	;	セミコロン
^	矢印またはべき乗記号	:	コロン
(	左カッコ	?	疑問符
)	右カッコ	<	より小さく
%	パーセント	>	より大きく
#	ナンバ記号	"	ダブルクォーテーション
$	ドル記号	_	アンダスコア

3. 予約語

予約語はBASICであらかじめその意味および用途が定められた言葉です。
予約語は命令の名前、関数名および演算子に用いられます。

● 予約語一覧

BASICにおける予約語の一覧を以下に示します。

ABS	CONNECT		IF
AKCNV$	CONSOLE	EDIT	IMP
AND	CONT	ELSE	INKEY$
ASC	COS	END	INP
ATN	CSNG	ENDIF	INPUT
AUTO	CSRLIN	EOF	INPUT$
	CVD	EQV	INSTR
BAUD	CVI	ERASE	INT
BEEP	CVL	ERL	INTERVAL
BGM	CVDMBF	ERR	
	CVS	ERROR	JIS
CALL	CVSMBF	EXEC	JTRIG
CDBL		EXP	JPAD
CD	DATA		
CDINF	DATE	FIELD	KACNV$
CDSTAT	DATE$	FILES	KANJI
CDSTIME$	DEF	FIX	KEXT$
CHAIN	DEFDBL	FN	KEY
CHR$	DEFINT	FOR	KILL
CINT	DEFLNG	FRE	KINSTR
CIRCLE	DEFSNG		KLEFT$
CLEAR	DEFSTR	GCURSOR	KLEN
CLNG	DELETE	GET	KMID$
CLOSE	DIM	GO	KNJ$
CLS	DRAW		KRIGHT$
COLOR	DSKF	HARDC	KTYPE
COM	DSKI$	HEX$	
COMMON	DSKINI		LEFT$

1



予約語の前後は、1文字以上の空白または特殊記号で切り離す必要があります。

LEN	NOT	REM	SWAP
LET		RENUM	SYMBOL
LINE	OCT$	RESET	STSTEM
LIST	OFF	RESTORE	
LLIST	ON	RESUME	TAB
LOAD	OPEN	RETORN	TALK
LOADM	OR	RIGHT$	TAN
LOC	OUT	RND	TERM
LOCATE	OUTM	ROLL	THEN
LOF	OUTPUT	RSET	TIME
LOG		RUN	TIME$
LPOS	PAD		TO
LPRINT	PAINT	SAVE	TROFF
LSET	PALETTE	SCREEN	TRON
	PART	SEARCH	
MAP	PASTEL	SGN	UNLIST
MERGE	PCMPLAY	SHELL	USING
MID$	PCMREC	SIMPOSE	USR
MIKD$	PEEK	SIN	
MKDMBF$	PEN	SINPUT	VAL
MKI$	PLAY	SKIPF	VARPTR
MKL$	POINT	SMSGPLAY	VIEW
MKS$	POKE	SOUND	VOICE
MKSMBF$	POS	SPACE$	
MOD	PRESET	SPC	WAIT
MON	PRINT	SPRITE	WEND
MOUSE	PSET	SQR	WHILE
MOVIE	PTRIG	STEP	WIDTH
	PUT	STOP	WINDOW
NAME		STR$	WRITE
NEW	RANDOMIZE	STRING$	
NEXT	READ	SUB	XOR

● 省略形

命令を表す予約語のうち次のものは省略形があり、命令を入力するときにこれらの省略形を使用することができます。省略形を使用して入力したプログラムも画面再表示および印刷時は完全な形で出力されます。

予約語	省略形
CONSOLE	CONS.
CONT	C.
COLOR	COL.
GOSUB	GOS.
GOTO	GO.
LOCATE	LOC.
LIST	L.
PRINT	?
PRINT#	? #
PRINT USING	? USING
RANDOMIZE	RNDM.
RETURN	RET.
REM	'
RUN	R.
WIDTH	W.

予約語と同じつづりの変数名は使用できません。予約語と同じつづりの変数名を使用した場合、BASICはそれを予約語として解釈するので、エラーが発生したり意図したとおりに動作しなかったりします。

4. 定 数

定数は、それ自身が値を表します。

BASICで扱える定数には文字定数と数値定数があります。

4.1 文字定数

文字定数はBASICで扱える文字セットを並べた文字列です。文字列の前後をダブルクォーテーション(”)で囲って記述します。

後のダブルクォーテーション(”)は省略することができます。このとき、行末までの文字列が文字定数として扱われます。

ただし、DATA文で文字定数を定義するときには、コンマ(,)、コロン(:)または空白を含まない場合はダブルクォーテーション(”)で囲む必要はありません。

文字数は、ANK文字で255文字、日本語(JIS第1水準・第2水準・非漢字)では127文字以下でなければなりません。

特に、文字列の長さが0のものを「空文字列」といいます。

記述例：

```
"THANK YOU"
"ヒンメイ　タンカ　ウリアゲ"
"漢字コード"
""        (空文字列)
```

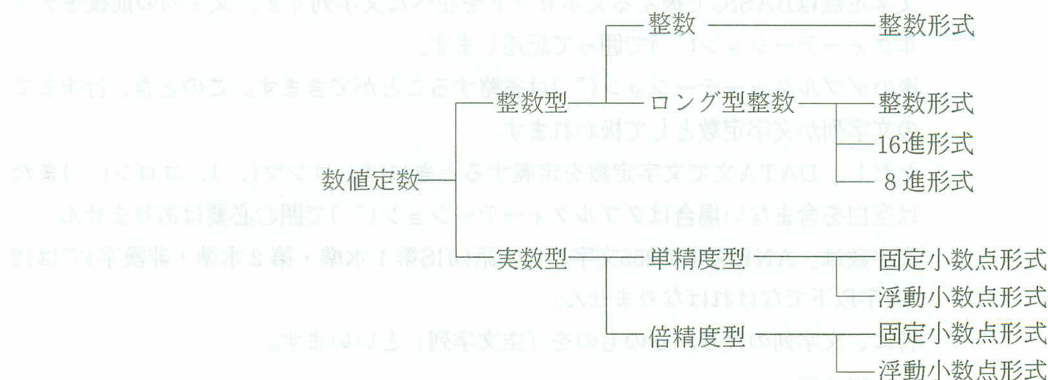
文字定数の型を「文字型」と呼ぶことがあります。

文字定数———文字型

4.2　数値定数

数値定数は正または負の数、または0です。BASICでは数値定数の間にコンマを含めることはできません。

数値定数には、次のような型があります。



●　整数型

(1)　整数形式

プラス符号の付いたものを正の整数、マイナス符号の付いたものを負の整数と呼び、プラス符号は省略できます。整数形式には整数とロング型整数があります。

- ●整数　　　　：－32768～＋32767の範囲の値で、数値の後に％を付けます。
- ●ロング型整数：－2147483648～＋2147483647の範囲の値で、数値の後に＆を付けます。

記述例：

```
1
+123%
2315479015&
```

(2) 16進形式

16進数を表す記号の&Hに続いて、16進数(0～9, A～F)を並べて書きます。

先行する0を除いて8桁を書くことができ、&H0～&HFFFFFFFFまでの範囲を表すことができます。

この形式で入力された16進数は、符号なし10進数に変換されて出力されます。

記述例：

```
100 A16=&H003FF
110 PRINT A16
```

実行結果

```
 1023
```

(3) 8進形式

8進数を表す記号の&Oまたは&に続いて8進数(0～7)を並べて書きます。

先行する0を除いて11桁を書くことができ、&O0～&O37777777777までの範囲を表すことができます。

この形式で入力された8進数は、符号なし10進数に変換されて出力されます。

記述例：

```
100 A08=&O01777
110 PRINT A08
```

実行結果

```
 1023
```

● 実数型

(1) 単精度型

単精度型の数値定数は、7桁までの精度で格納され、6桁までの桁数で表示されます。

単精度定数で表現できる数値の範囲は、−3.40282E+38〜+3.40282E+38までであり、−1.17549E〜−38〜+1.17549E−38の範囲の値はすべて0になります。

単精度数値定数は、次のいずれかに当てはまるものです。

・有効桁数が7桁以下の定数
・Eを用いた指数形式で表現されたもの
・最後にエクスクラメーション(！)が書かれた定数

記述例：

```
13.6
-1.23E+6
862.1!
520
```

(2) 倍精度型

倍精度型の数値定数は、16桁までの精度で格納され、15桁までの桁数で表示されます。

倍精度定数で表現できる数値の範囲は、−1.79769313486231D+308〜+1.79769313486231D+308までであり、−2.2250738585072D−308〜+2.2250738585072D−308の範囲の値はすべて0になります。

倍精度数値定数は、次のいずれかに当てはまるものです。

・有効桁数が8桁以上の定数
・Dを用いた指数形式で表現されたもの
・最後にナンバ記号(＃)が書かれた定数

記述例：

```
1368797202
-1.5670D-12
562.983#
852.138269012
```

(3) 固定小数点形式

符号に続いて、整数部、小数点、小数部の順に書いたもので、プラス符号は省略することができます。

整数部と小数部のうち、一方は省略できますが、両方とも省略することはできません。

小数点はピリオド(.)で表します。

数値の後に！をつけると単精度の数値定数に、#をつけると倍精度の数値定数になります。

記述例：

```
1.0
-123.11
.999
```

(4) 浮動小数点形式

指数形式で表現された正または負の数値で、符号に続いて、整数部、小数点、小数部、指数部の順に書いたものです。プラス符号は省略することができます。

整数部、小数部のうち、一方の省略あるいは、小数点と小数部の省略ができます。

指数部は、単精度ではE〔{±}〕nnで表し、倍精度ではD〔{±}〕nnnで表します。

なお、nnやnnnは符号なしの整数です。

浮動小数点形式では、数値のあとに！や#を付けてはいけません。

記述例：

```
520.18E+7
108E-18
3.1256E+12
9.99999999D-38
```

5. 変　数

変数は、BASICのプログラムの中で使用される値を表すために使われる名前です。
定数と同じように、変数にも数値変数と文字変数の2つがあります。

● **文字変数**

文字列だけを持つ変数です。文字変数の長さは、値が代入されたときに決まります。何も代入されないときは空文字列(長さ0の文字列)になっています。

● **数値変数**

常に数値を持つ変数です。変数に値を与える前は0になっています。

変数に値を与えるためには、代入文、INPUT命令、READ命令などを用います。
いずれの場合も変数の型は、それに割り当てられる定数の型と一致していなければなりません。

5.1　変数名

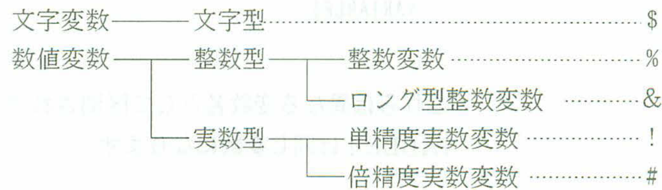
任意の日本語文字、ANK文字の英数字およびアンダースコア(_)を続けて変数名とします。
ただし、先頭はANK文字の英字または日本語文字でなければなりません。
変数名においてはANK文字の英字の大文字と小文字の区別はありません。小文字で指定した文字は、大文字に変換されてメモリに格納されます。
また、日本語と英数字の混在も許されます。ただし、インタプリタでは、日本語を含む変数名は128個まで使用できます。コンパイラでは、日本語を含む変数名の個数に制限はありません。
変数名の長さは、それを含む文が1行の制限文字数以内であれば、何文字であってもかまいません。ただし、変数名が40文字を超えた場合、インタプリタでは、最初の40文字と型宣言文字により変数名を区別します(型宣言文字は、省略可能です)。
コンパイラでは、すべての文字で変数名を区別します。
変数名は予約語であってはいけません。

記述例：

```
LINE  ←  予約語であり変数名には使用できません.
LINEA ←  変数名として許されます.
ALINE ←  変数名として許されます.
```

5.2 変数の型

変数名の直後に型宣言文字を続けて、その変数の型を宣言することができます。この型宣言文字を使用することにより、変数名が同じでも型が違うことで別の変数として扱うことができます。型宣言文字は、ANK文字で記述し、次のものがあります。



型宣言文字	意　　味
$	文字変数を示し、ANK文字で255文字、日本語文字(JIS第1水準・第2水準・非漢字)で127文字まで格納することができます。
%	整数変数を示し、1つの変数につき2バイトのデータ格納域を必要とします。
&	ロング型整数変数を示し、1つの変数につき4バイトのデータ格納域を必要とします。
!	単精度変数を示し、1つの変数につき4バイトのデータ格納域を必要とします。
#	倍精度変数を示し、1つの変数につき8バイトのデータ格納域を必要とします。

変数の型を宣言する方法としてほかに型宣言命令(DEFINT, DEFLNG, DEFSNG, DEFDBL, DEFSTR)があります、ただし、型宣言文字の指定の方が優先されます。
型宣言命令もなく型宣言文字を伴わない変数名は、単精度形式の数値変数として扱われます。

記述例1：変数名の直後に型宣言文字を付けます。

```
L$      文字変数
A%      整数変数
BSC&    ロング型整数変数
PNT!    単精度変数
MAX#    倍精度変数
AB      単精度変数 —— 型宣言文字を省略した場合
```

記述例２：変数名が同じでも型が違うと、別の変数として扱われます。

```
VARIABLE%
VARIABLE&
VARIABLE!
VARIABLE#
VARIABLE$
```

これらは異なる変数名として区別されますが、ＶＡＲＩＡＢＬＥとＶＡＲＩＡＢＬＥ！は同じ変数になります。

5.3 配列変数

配列は、同じ性質を持つ複数個のデータの集まりです。配列は同一名でその要素を参照することができ、それぞれの要素は添字により順序づけられます。
ある変数を配列変数として宣言するためには、DIM命令を用いて次のように行います。

● 配列変数の定義

DIM　**変数名(添字の最大値，添字の最大値…)**

カッコの中に書かれた「添字の最大値」の個数により、その配列変数の次元数を指定します。配列の次元は、1次元から多次元の指定が行えます。また、各次元の添字の最大値が10を超えなければ、DIM文なしに配列変数の要素を参照することができます。

● 配列変数の参照

配列変数の要素の参照は、各次元の添字を指定した添字付き名で行い、その形式は次のとおりです。

変数名(第1次元の添字，第2次元の添字…)

変数名の直後にカッコでくくって各次元の添字を指定します。この中に書かれた添字の個数は、配列変数の次元数と一致しなければなりません。添字は数値式で、その値は0から添字の最大値までです。各次元の添字が整数でない場合は、小数部を四捨五入した整数またはロング型整数に変換されます。

記述例 1 : 1 次元の配列

配列の定義… DIM A$(5)

配列の要素…

A$(0)
A$(1)
A$(2)
A$(3)
A$(4)
A$(5)

要素数　6

記述例 2 : 2 次元の配列

配列の定義… DIM B(2,3)

配列の要素…

行 ＼ 列			
B(0,0)	B(0,1)	B(0,2)	B(0,3)
B(1,0)	B(1,1)	B(1,2)	B(1,3)
B(2,0)	B(2,1)	B(2,2)	B(2,3)

要素数　12

メモリ上での並びは次のようになっています.

B(0,0)
B(1,0)
B(2,0)
B(0,1)
B(1,1)
B(2,1)
B(0,2)
B(1,2)
B(2,2)
B(0,3)
B(1,3)
B(2,3)

6.　型変換

BASICの文に書かれた数値データは、実行のとき必要に応じて、自動的にその型から他の型に変換されます。変換が行われるのは、次のような場合です。

●異なる型の数値変数に代入したとき。

●精度の異なる変数間で演算したとき。

●同じ型同士の演算結果が、その型で表せる範囲を超えたとき。

以下にその変換の規則について述べます。

●異なる型の数値変数に代入したとき

(1)　実数データを整数変数に代入したとき

小数点以下が四捨五入され、整数値に変換されます。このとき、変換されたデータが整数型で表せる範囲を超えているときは、エラーになります。

例 1

```
10 A%=12.34
20 B%=35.56
30 PRINT A%,B%
```

実行結果

```
 12          36
```

例 2

```
10 A%=1.23E+06
20 PRINT A%
```

実行結果

"数値データの値が許される範囲を超えています"
というエラーメッセージが表示されます。

以上のような実数値から整数値への変換は、代入文のみならず、関数および文の評価のときにも行われます。具体的には、以下のような場合です。

・配列要素の添字が実数のとき

・キャラクタコードが実数のとき

・ファイル番号が実数のとき

・PFキーの番号が実数のとき

・オペランドに整数値を要求する文で、実数が用いられたとき

・引数に整数値を要求する関数の引用において、実数が用いられたとき

(2) より精度の低い変数に代入したとき

四捨五入が行われます。たとえば、倍精度の数値データを単精度変数に代入するような場合です。

例：

```
10 A=12.3456789#
20 PRINT A
```

実行結果

```
 12.3457
```

以上のような変換は、代入文のみならず、次のような場合にも行われます。

・ワールド座標の値として倍精度実数が指定された場合、単精度実数に変更して処理されます。

(3) より精度の高い数値変数に代入したとき

その変数の型に変換後代入されます。しかし、結果として得られる高精度の数値は、元のデータよりも精度が高くなることはありません。たとえば、Bという単精度のデータをA#という倍精度変数に代入したとすると、A#の最初の6桁のみ正しい精度が得られます。これは、Bというデータ自体が6桁の精度しか持っていないからです。

この変換における相対誤差は、次の式で表すことができます。

相対誤差＝ABS(A#−B)/B)<1.71E−06

すなわち、倍精度の数値と元の単精度の数値との差を元の単精度の数値で割った値の絶対値は、1.71E−06より小さくなります。

例：

```
10 B=3.02
20 A#=B
30 PRINT B,A#
```

実行結果

```
 3.02          3.01999998092651
```

1

● 精度の異なる変数間で演算したとき

(1) 算術演算および関係演算の場合

低い精度を高い精度に変換後、同じ精度で演算が行われます。したがって、演算結果は高い方の精度になります。

例１：7#/６の除算は倍精度で行われ、その結果が倍精度でA#に返されます。

```
10 A#=7#/6
20 B#=7#/6#
30 PRINT A#
40 PRINT B#
```

実行結果

```
 1.16666666666667
 1.16666666666667
```

例２：7#/６の除算は倍精度で行われます。Ａは単精度変数であるため、その結果が単精度に丸められてＡに代入されます。

```
10 A=7#/6
20 PRINT A
```

実行結果

```
 1.16667
```

(2) 論理演算の場合

扱う数値はすべて小数部を四捨五入することにより、ロング型整数に変換され演算が行われます。ロング型整数値に変換したとき、その値は−2147483648～+2147483647の範囲になければなりません。結果は整数値(−2147483648～+2147483647)で返されます。

例1：

```
10 A#=34.55
20 B=NOT A#
30 PRINT B,A#
```

実行結果

```
-36          34.5499992370605
```

例2：

```
10 A#=34.55D+170
20 B=NOT A#
30 PRINT B,A#
```

実行結果

"数値データの値が許される範囲を超えています"というエラーメッセージが表示されます。

● 同じ型同士の演算結果が、その型で表せる範囲を超えたとき演算結果は、より高い精度の変数に変換されます。

例：

```
10 A&=2147483647&
20 B&=2147483647&
30 PRINT A&+B&
```

実行結果

```
4294967294
```

7. 式

1

式は演算を実行するものであり、定数や変数、関数を演算子で結んで表します。
式の演算結果は、1個の数値または1個の文字列なので、単に数値や文字、あるいは変数のみであっても「式」と呼びます。式を分類すると、次の4種類になります。

・文字式
・算術式
・関係式
・論理式

これらの中で、算術式、関係式および論理式は、その評価結果が数値であることからまとめて数値式と呼びます。

7.1 文字式

文字変数または文字定数を演算子プラス(+)によって連結した式を文字式と呼びます。文字式で用いられる演算子のプラス(+)は、二つの文字データの加算ではなく、二つの文字列を連結することを意味します。また、単に一つの文字変数、または文字定数も文字式と呼びます。

例：
```
10 A$="FILE"
20 B$="NAME"
30 PRINT A$+B$+"="
```

実行結果
```
FILENAME=
```

7.2 算術式

算術式は、変数、定数、関数などの数値データを算術演算子で結んだものです。また、算術演算子のないデータのみの場合も算術式と呼びます。

算術演算子とその意味は次のとおりです。

算術演算子	内　　容	記述方法
+	加算を行います。	A + B
−	減算を行います。	A − B
*	乗算を行います。	A * B
/	除算を行います。	A / B
^	べき乗を行います。	A ^ B
¥	整数の除算を行います。 結果は浮動小数点除算の小数点以下を切り捨てた値となります。	A ¥ B
MOD	整数の除算の剰余を求めます。	A　MOD　B

整数の除算(¥)および整数の剰余(MOD)の演算において、扱う数値が整数形式でないときは、小数部分を四捨五入し、整数に変換してから演算が行われます。

例：

```
24.35¥6.87=3     (24¥7)
10.2 MOD 4=2     (10MOD4)
```

7.3 関係式

関係式は数値または文字列の２つの値を比較するときに用い、比較の結果、真ならー１が、偽ならば０が設定されます。関係式は、主にIF命令やWHILE命令などで、プログラムの流れを変える場合に用いられます。関係演算子とその意味を以下に示します。

関係演算子	内　容	表　記
＝	等しい	A＝B
<>、><	等しくない	A<>B
＜	小さい	A＜B
＞	大きい	A＞B
<=、=<	等しいか小さい	A<=B
>=、=>	等しいか大きい	A>=B

● 数値の比較

数値の大小を比較します。

例１：

```
10 A=10:B=20
20 IF A>B THEN PRINT "A>B"
          ELSE PRINT "A<B"
```

実行結果

```
 A<B
```

例２：

```
10 A=10<20:B=10>20
20 PRINT  A,B
```

実行結果

```
-1             0
```

● 文字列の比較

文字列の比較は、先頭文字より1文字ずつ、対応するキャラクタコードで比較します。すべてのキャラクタコードが等しければ、文字列は等しくなります。キャラクタコードが等しくなければ、小さいキャラクタコードの文字列は大きいキャラクタコードの文字列よりも小さいという結果になります。もし、比較の途中で文字列が終わりになったら、短い文字列が小さいという結果になります。文字列の中の空白も比較の対象になりますので注意してください。

例：

```
"ABC"="ABC"
"ABC"<"ABD"
"A&">="A#"
"PR ">"PR"
"cm">"CM"
"LOAD"<"LOADM"
A$<"02:37:15"(A$="01:25:48" のとき)
```

7.4 論理式

ビット操作や論理演算を行ったり、いくつかの関係式を調べたりするときに、論理式を用います。論理式は、論理演算子と変数、定数、関数などの数値データの結合により構成されます。
なお、数値データが整数でないときは、小数点以下の四捨五入した整数に変換されます。

● 論理演算子

論理演算子にはNOT, AND, OR, XOR, IMP, EQVがあります。
各論理演算では、ビットごとに次の結果を与えます。

・NOT(否定)

X	NOT X
1	0
0	1

・AND(論理積)

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

・OR(論理和)

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

・XOR(排他的論理和)

X	Y	X　XOR　Y
1	1	0
1	0	1
0	1	1
0	0	0

・IMP(包含)

X	Y	X　IMP　Y
1	1	1
1	0	0
0	1	1
0	0	1

・EQV(同値)

X	Y	X　EQV　Y
1	1	1
1	0	0
0	1	0
0	0	1

論理演算子も関係演算子のように、プログラムの流れを変えるのに用いられます。

例：

```
10  X=30
20  IF X>0 AND X<999 THEN 300
```

● 論理演算

論理演算では、整数に対しビットごとに演算を行います。

例：論理演算の計算例を示します。

・ＮＯＴ(否定)

A%=NOT 1
1=$(0000000000000001)_2$ によりNOT 1=$(1111111111111110)_2$=-2

・ＡＮＤ(論理積)

B%=2 AND 3
2=$(0000000000000010)_2$, 3=$(0000000000000011)_2$ より
B%=$(0000000000000010)_2$=2

・ＯＲ(論理和)

C%=-1 OR -4
-1=$(1111111111111111)_2$, -4=$(1111111111111100)_2$ より
C%=$(1111111111111111)_2$=-1

・ＸＯＲ(排他的論理和)

D%=4 XOR -3
4=$(0000000000000100)_2$, =-3$(1111111111111101)_2$ より
D%=$(1111111111111001)_2$=-7

・ＩＭＰ(包含)

E%=1 IMP 3
1=$(0000000000000001)_2$, 3=$(0000000000000011)_2$ より
E%=$(1111111111111111)_2$=-1

・ＥＱＶ(同値)

F%=5 EQV -3
5=$(0000000000000101)_2$, =-3$(1111111111111101)_2$ より
F%=$(0000000000000111)_2$=7

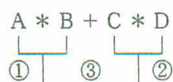
7.5 演算子の優先順位

優先順位	演算子
1	^
2	−(負記号), +(正記号)
3	*,/
4	¥
5	MOD
6	+,−
7	関係演算子
8	NOT
9	AND
10	OR
11	XOR
12	EQV
13	IMP

式の中の演算は、優先順位の高い演算子から実行され、同じ優先順位のときは左から実行されます。演算子の優先順位を次に示します。

かっこ(　)で囲まれた式および関数は、演算子の優先順位に関係なく先に行われます。

例１：



例２：



8. ファイル

8.1 ファイルディスクリプタ

ファイルディスクリプタとは、入出力装置、ファイルおよびレコード長(オプション)を指定するための名前です。以下に、ファイルディスクリプタにより、指定することのできる項目を示します。

・入出力命令の中には、入出力装置を自由に指定できない命令があります。
たとえば次の命令では、デバイス名は固定です。

```
INPUT ························KYBD:
LPRINT        ⎫
LPRINT USING  ⎪
LLIST         ⎬······LPT0:
HARDC         ⎭
```

● ファイルディスクリプタの形式

ファイルディスクリプタは、次の形式の文字列により、構成されます。

"デバイス名　(オプション)　パス名　ファイル名"

・デバイス名、オプションおよびファイル名は、それぞれ省略できる場合があります。
・ディスク以外の場合には、パス名とファイル名を指定しても意味を持ちません。
・ディスクの場合には、デバイス名とパス名は省略できますが、FILES命令を除きファイル名は指定しなければなりません。
・オプションは命令によって指定するものとそうでないものがあります。
・ファイルディスクリプタは、全体をダブルクォーテーション(")で囲って記述しますが、文字変数、または文字式であってもかまいません。

(1)　デバイス名

デバイス名は、そのファイルの存在する物理装置名を示します。

BASICで使用できる入出力装置のデバイス名は次のとおりです。

入出力装置名		デバイス名	入力	出力	備　　考
キーボード		KYBD :	○	×	
スクリーン		SCRN :	×	○	
プリンタ	0	LPT0 :	×	○	
	1	LPT1 :			
	2	LPT2 :			
	3	LPT3 :			
RS-232C	0	COM0 :	○	○	
	1	COM1 :	○	○	
	2	COM2 :	○	○	
	3	COM3 :	○	○	
	4	COM4 :	○	○	
ディスクドライブ番号	0	0 :	○	○	0 :～ 9 : はMS-DOSのドライブ名A:～J:に対応します。
	～	～			
	9	9 :			
MS-DOSのドライブ名		A :			ただし、CD-ROMドライブには出力できません。
		～			
		Q :			



デバイス名は、小文字では指定できません。

MS-DOSのドライブ名は、小文字で指定しても大文字として扱われます。

(2)　オプション

〈オプション〉は、ランダムファイルのOPEN命令でレコード長を指定します。

ランダムファイルのレコード長は、1～4桁の数字でバイト数を指定します。指定できる値の範囲は1～2048で、省略すると256になります。

(3) パス名

〈パス名〉は、目的のファイルに達するまでの経路を示します。
指定できるパス名の最長は63文字で、以下の形式で指定します。

〔¥〕〔ディレクトリ名¥〕…〔ディレクトリ名¥〕

先頭の円記号〔¥〕は、ルートディレクトリを表し、省略した場合にはカレントディレクトリからのパスとなります。

ディレクトリ名には親ディレクトリを示す“..”を用いることもできます。
パス名全体を省略した場合には、カレントディレクトリとなります。パス名は小文字で指定しても大文字として扱われます。

(4) ファイル名

〈ファイル名〉は、ファイルにつけられた名前であり、〈デバイス名〉で示される物理装置の中のどのファイルであるかを示します。
ファイル名は次の形式で記述します。

ファイル名〔.〔拡張子〕〕

拡張子は一般にファイルの型を表すために使用します。
BASICのプログラムを保存するときに拡張子を指定しないと、“BAS”が拡張子として付けられます。プログラムを読み込んだり、マージしたり、CHAINコマンドで連結するときにも、拡張子を指定しないと、“～.BAS”のファイルを指定したことになります。保存や読み込み、MERGE, CHAINで特に拡張子のついていないファイル名を指定したい場合には、“.” までを指定します。

例）

```
CHAIN "MYPRG"      }→ファイル"MYPRG.BAS"を連結します.
CHAIN "MYPRG.BAS"  }
CHAIN "MYPRG."  ――――→ファイル"MYPRG"を連結します.
```

なお、データファイルについては省略時にこのような解釈は行われません。

ファイル名の規則は次のとおりです。

・ファイル名は8文字以内、拡張子は3文字以内。
・.BASは、BASICプログラムのファイルです。
・.EUPは、FMTOWNS標準形式の音楽演奏のデータファイルです。
・.JPGは、JFIFフォーマットのイメージファイルです。
・.MVEは、LiveMovieのデータファイルです。
・.MMMは、LiveAnimationのデータファイルです。
・.TIFは、FMTOWNS標準形式のイメージデータのファイルです。
・.PMBは、FMTOWNS標準形式のPCM音色のファイルです。
・.FMBは、FMTOWNS標準形式のFM音声のファイルです。
・.SNDは、FMTOWNS標準形式のサンプリングデータのデータファイルです。
・空文字列は使用できません。
・FILES命令でのみ、ワイルドカードが使用できます。
・コロン(:)、ダブルクォーテーション(")左カッコ(()、右カッコ())、スラッシュ(/)、アスタリスク(*)、疑問符(?)および16進数の00とFFは使えません。
・次の文字列は使用できません。

CON, PRN, AUX, CLOCK, NUL

> **用語**
> ワイルドカード
>
> ワイルドカードは、ファイル名の全ての文字を記述するかわりに使うふせ字(*と?のこと)です。
>
> *は任意の個数の任意の文字列、?は任意の1文字の数字を表します。
>
> ワイルドカードを使ったファイル名の指定のしかたは次のとおりです。
>
> FILES "A*.BA?"
>
> Aで始まり、拡張子が"BAx"(xは任意の1文字)のファイルを全て表示します。
>
> FILES "*.BAS"
>
> BASICプログラムのファイルを全て表示します。

● ファイルディスクリプタの記述例

"0:FILE1"	ドライブ番号0
"B:FILE2"	ドライブ名B
"1:MAIN.BAS"	ドライブ番号1
"LPT0:"	プリンタ
"COM0:"	RS-232C
"Q:FJ2¥TONE¥PRESET1"	CD-ROMドライブ

8.2 ファイル番号

BASICの入出力操作には、ファイル番号を用います。ファイル番号は、ファイルディスクリプタに対して割り当てられた番号で、OPEN命令によって定義され、ファイルディスクリプタとの対応づけが行われます。

このファイル番号は1から16の範囲の値で指定し、定数、変数、または式で表すことができます。

8.3 ファイル上のプログラム形式

メモリ上のBASICのプログラムは、ファイル制御ウィンドウの「保存」またはSAVE命令でファイルに格納できます。ファイルへの格納形式には、アスキー形式とバイナリ形式とがあります。

● アスキー形式

プログラムを構成する文字をすべて1文字ずつ、ASCIIコードで表す形式です。

MERGE機能を実行する場合には、プログラムはアスキー形式で格納されていなければなりません。また、BASICのエディタ以外で扱う場合には、アスキー形式でないと取り扱うことができません。

● バイナリ形式

プログラムを効率よく内部形式に変換して表す形式です。アスキー形式と比較してファイルの大きさが小さくてすみ、保存も読み込みも高速にできます。

9. 初期設定

BASIC起動時は、以下の状態に設定されています。
この状態に戻すには、右のような命令を実行します。

起動時の設定	初期化命令
グラフィック画面：1画面モード 画面モード　　：16色モード ビューポート　：(0, 0)-(639, 479) ウィンドゥ　　：(0, 0)-(639, 479)	SCREEN 0 SCREEN@ 0 WINDOW
テキスト画面　：80字×25行	WIDTH 80, 25
スクロールウィンドウ：0〜24行 PFキーの表示　：なし	CONSOLE 0, 25, 0
スプライト画面：OFF	SPRITE OFF
ビデオ画面　　：OFF	SIMPOSE OFF
文字色：白，背景色：黒，前景色：白	COLOR 7, 0, 7, 0
ペンの太さ：1(ドット)	DEF PEN 0, 1 ただし、次の命令を実行すると、初期化されます。 SIMPOSE SINPUT SPRITE ON/OFF VIEW
パレット　：	PALETTE
パステル　：128	PASTEL
スプライトキャラクタ数：0	DEF SPRITE 99, 0
音源　　　：システム標準の音色データ	LOAD@
音源チャンネル：パート0〜15に チャンネル0〜15が 割り当てられている。	FOR I=0 TO 15 PART I, I NEXT I
マウス　　：OFF	MOUSE 5

10.　BASICの作業領域

BASICは、メモリの作業領域を6つの目的に分けて使用します。

● テキスト領域

プログラムテキスト格納領域として使用します。
この領域の大きさは、全作業域から、配列変数領域、スタック領域、DLL領域、プロシジャ領域および内部作業域を除いた大きさです。

● 単純変数領域

単純変数テーブル、文字領域、ファイル用バッファとして使用します。
単純変数領域は、テキスト領域の中の未使用領域が自動的に割り当てられますので、CLEAR命令により単純変数領域の大きさを指定することはできません。
この領域の大きさを変更したい場合は、CLEAR命令で配列変数領域やスタック領域等の大きさを変更してください。

● 配列変数領域

配列変数テーブルとして使用します。この領域の大きさは、CLEAR命令により、作業領域の範囲内で可能な限り指定できます。BASIC起動時には、スタック領域を除いた残りの1/3が配列変数領域となります。

● スタック領域

この領域の大きさは、CLEAR命令により指定できますが、256バイトは必ず必要です。BASIC起動時には、512バイト確保されます。

● プロシジャ領域

プロシジャ(機械語プログラム)を読み込んで実行するための領域として使用します。この領域の大きさは、CLEAR命令によって指定します。
BASIC起動時には、この領域はありません。

● DLL領域

DLL(ダイナミックリンクプログラム)を読み込んで実行するための領域として使用します。この領域の大きさは、CLEAR命令によって指定します。BASIC起動時には、この領域はありません。

第2章

命令の概要と使い方

命令を分類し、概要を説明します。
特に、いくつかの命令を組み合わせて使うものについては、その順序と意味を説明しています。

1. 目的別命令分類表

ここでは命令を用途別に分類しています。説明が必要なものは次の節以降で概要を説明します。

(1) 画面表示

テキスト画面の設定や表示に関する命令

用　　途	命　　令
画面の行数、文字数を設定する	WIDTH, CONSOLE
文字を表示する	PRINT, PRINT USING, WRITE, TAB, SPC
画面上の位置を指定する	LOCATE, CSRLIN, POS
文字の色を指定する	COLOR
外字を設定する	DEF KANJI
画面をクリアする	CLS

グラフィック画面の設定や描画に関する命令

用　　途	命　　令
画面の重ね合わせを変更する	SCREEN@, SCREEN
ウィンドウの範囲を定義する	WINDOW, WINDOW関数
ビューポートの範囲を定義する	VIEW, VIEW関数
VRAMをスクロールする	ROLL
最終参照座標を変更する	POINT
座標値を変換する	MAP関数
背景色、前景色を設定する	COLOR
パレットの色を設定する	PALETTE
ペンの太さと形状を設定する	DEF PEN
描画する	LINE, CONNECT, CIRCLE, PSET/PRESET, PAINT
色を塗る	PUT@, GET@
文字を表示する	SYMBOL, DEF FONT
色の混合比率を設定する	PASTEL
画面をクリアする	CLS

スプライト画面の設定や操作に関する命令

用　　途	命　　令
画面を設定する	SPRITE ON/OFF, SPRITE SCREEN
スプライトを定義する	DEF SPRITE
スプライトを操作する	SPRITE, SPRITE関数, SPRITE TIME

ビデオ画像の表示や操作に関する命令

用　　途	命　　令
ビデオ画面を表示する	SIMPOSE ON/OFF
画像をVRAMに読み込む	SINPUT

(2) 印刷

プリンタで印刷するための命令です。書式制御文字で印刷形式（書式）を指定することができます。

用　　途	命　　令
結果を印刷する	LPRINT, LPRINT USING, WRITE#, PRINT#
印字桁位置を知る	LPOS, POS
印字行数を設定する	WIDTH
タブ機能、スペース印字	TAB, SPC
画面を印刷する	HARDC

画面表示の命令と、次のように対応しています。

PRINT －－－－－－－－－－LPRINT
PRINT USING －－－－－－LPRINT USING
WRITE －－－－－－－－－－－WRITE#
POS －－－－－－－－－－－－LPOS

(3) データ入力

用　　途	命　　令
定数を定義する	DATA
定数を変数に読み込む	READ, RESTORE
キーボードから入力する	INPUT, LINE INPUT, INKEY$, INPUT$

(4) データファイル

用　　途	命　　令
ファイルを開く、閉じる	OPEN, CLOSE
シーケンシャルファイルから読む	INPUT#, LINE INPUT#, INPUT$関数
ランダムファイルから読む	GET
シーケンシャルファイルに書く	PRINT#, WRITE#
ランダムファイルに書く	PUT
ファイルの終りを検出する	EOF関数, LOF関数
ファイル名を変更する	NAME
ファイル名を削除する	KILL
ファイル名を一覧表示する	FILES
バッファを定義する	FIELD
バッファに代入する	LSET/RSET
レコード位置を確認する	LOC関数
文字列を数値に変換する	CVI/CVD/CVS/CVL, CVSMBF/CVDMBF
数値を文字型に変換する	MKI$/MKD$/MKS$/MKL$, MKSMBF$/MKDMBF$

(5) マウス／パッド

用　　途	命　　令
マウス機能を設定する	MOUSE
マウスカーソルの情報を知る	MOUSE関数
マウスによる割り込み処理	ON MOUSE (n) GOSUB, MOUSE (n) ON/OFF/STOP
パッドのボタンの状態を知る	PTRIG関数
パッドの方向を知る	PAD関数

(6) 文字列処理

用途	命令
文字の種類を調べる	KTYPE
定数文字列を作る	STRING$, SPACE$
文字列の長さを調べる	LEN/KLEN
文字列を検索する	INSTR/KINSTR
文字列を抽出する	LEFT$/KLEFT$, RIGHT$/KRIGHT$, MID$/KMID$, KEXT$
文字・数字変換	STR$/VAL
文字コードを変換する	AKCNV$/KACNV$, CHR$/ASC, JIS/KNJ$
数値を文字型に変換する	MKI$/MKS$/MKD$/MKL$, MKSMBF$/MKDMBF$
文字列を数値に変換する	CVI/CVD/CVS/CVL, CVSMBF/CVDMBF

(7) 計算

用途	命令
型を変換する	CINT/CSNG/CDBL/CLNG
結果を代入する	LET, LSET/RSET
数値関数	SQR, LOG, EXP, FIN, INT, ABS, SGN, RND
三角関数	SIN, COS, TAN, ATN
16進・8進変換	HEX$, OCT$
変数の値を交換する	SWAP

(8) PFキー

キーボード上に12個のPFキーがあります。このうち10個のPFキーに文字列を定義し、そのキーを押すと、その文字列が入力されるように設定できます。

用途	命令
キーに文字列を割り当てる	KEY
割り当てられた文字列を表示する	KEY LIST
PFキーで割り込む (PFキー割り込み)	KEY (n) ON/OFF/STOP, ON KEY (n) GOSUB

(9) 音楽／音声

用　途	命　令
ブザーを鳴らす	BEEP
音楽を演奏する	PLAY, PLAY@, PLAY関数, PLAY ON/OFF/STOP
並行動作を制御する	BGM
パートを割り当てる	PART, PART関数
音色を設定する	VOICE, VOICE SET, LOAD@
音色データを配列にコピーする	VOICE COPY
音声を記録する・再生する	PCMREC, PCMPLAY
MIDIポートへデータを送る	OUTM
サウンドメッセージを鳴らす	SMSGPLAY

(10) CD演奏

用　途	命　令
CDを演奏する	CD PLAY, CD STOP/PAUSE/CONT
CD-ROMドライブを調べる	CDSTAT
CD-ROMの内容を調べる	CDINF
曲の開始時間を調べる	CDSTIME$

(11) 時計

用　途	命　令
日付を知る	DATE$, DATE
時刻を知る	TIME$, TIME
日付時刻を設定する	DATE$=, TIME$=
一定時間ごとに割り込む (インターバルタイマ割り込み)	INTERVAL, INTERVAL ON/OFF/STOP, ON INTERVAL GOSUB
指定した時間に割り込む (タイマ割り込み)	TIME, TIME ON/OFF/STOP, ON TIME GOSUB
プログラムを中断する	WAIT

2

(12) 動画／アニメーション

用　　途	命　　令
ムービーファイルを再生する	MOVIE OPEN, MOVIE PLAY, MOVIE CLOSE
ムービーファイルの情報を得る	MOVIE INFO
再生指示情報を設定する	DEF MOVIE

(13) プログラムの分岐

用　　途	命　　令
分岐する	ON〜GOTO, GOTO, IF〜THEN〜ELSE
サブルーチンを呼び出す	ON〜GOSUB, GOSUB
サブルーチンから復帰する	RETURN
割り込み処理ルーチンを呼び出す (回線入力割り込み) (PFキー割り込み) (マウス割り込み) (タイマ割り込み) (インターバルタイマ割り込み)	ON COM (n) GOSUB, COM (n) ON/OFF/STOP ON KEY (n) GOSUB, KEY (n) ON/OFF/STOP ON MOUSE (n) GOSUB, MOUSE (n) ON/OFF/STOP ON TIME (n) GOSUB, TIME (n) ON/OFF/STOP ON INTERVAL (n) GOSUB, INTERVAL (n) ON/OFF/STOP
割り込み処理ルーチンから復帰する	RETURN
繰り返し実行する	WHILE〜WEND, FOR〜NEXT
IFブロック	IF〜THEN, ENDIF, ELSE, ELSE IF〜THEN
プログラムを連結する	CHAIN, COMMON
プログラムを終了する	END, SYSTEM
プログラムを中断する	STOP, STOP ON/OFF

(14) エラー処理ルーチン

用　　途	命　　　　令
エラー処理ルーチンを定義する	ON ERROR GOTO
エラー処理ルーチンから復帰する	RESUME
エラー発生行とエラー番号を知る	ERR/ERL
エラー発生をシミュレートする	ERROR

(⇨「付録1　F-BASIC386エラーメッセージ一覧」P.480)

(15) プログラムを操作する命令

プログラムそのものを扱う命令です。これらの機能はBASICエディタのメニューからも要求できます。また、エディタの「一行実行」で命令を入力して要求することもできます。プログラム中に書いた場合、SAVE以外は、その命令を実行するとプログラムが終了し、エディタに戻ります。

用　　途	命　　　　令
行を削除する	DELETE
プログラムを表示・印刷する	LIST, LLIST
プログラムをメモリに読み込む	LOAD
2つのプログラムを混ぜ合わせる	MERGE
メモリ上のプログラムを消去する	NEW
行番号を付け替える	RENUM
プログラムを実行する	RUN
プログラムをファイルに格納する	SAVE

(16) その他の命令

用　　途	命　　　　令
変数を初期化する	CLEAR
関数を定義する	DEF FN
配列変数を定義する	DIM
配列変数をメモリから消去する	ERASE
変数の型を定義する	DEFINT, DEFLNG, DEFDBL, DEFSNG, DEFSTR
コメントを書く	REM
配列変数の中から指定の値を探す	SEARCH
プログラムの実行を追跡する	TRON, TROFF

2. 画面表示

2

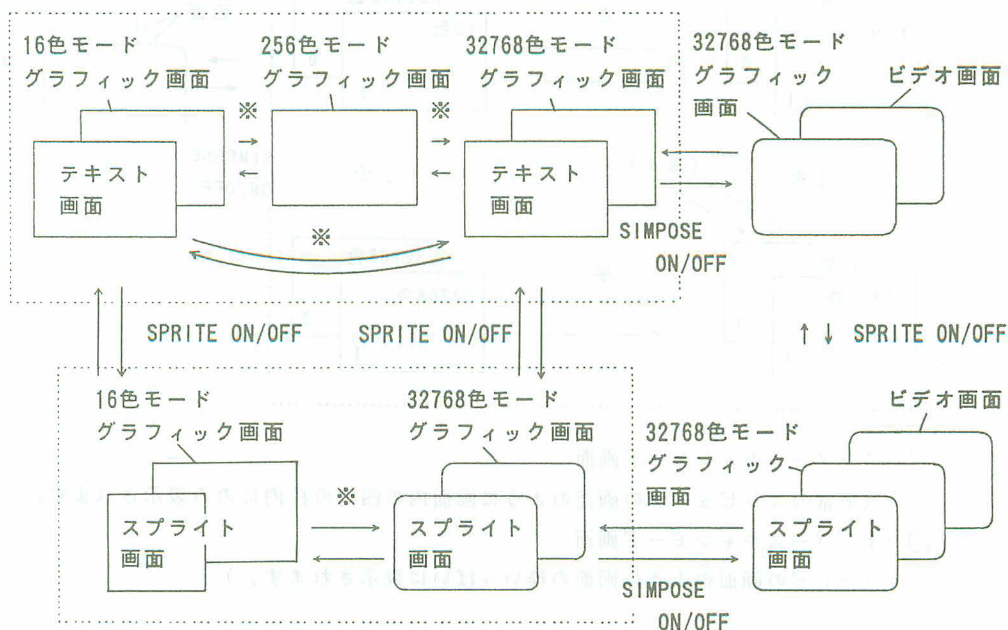
実行画面として次の4つの画面があり、2つまたは3つの画面を重ね合わせてディスプレイに表示します。

・テキスト画面
・グラフィック画面
・スプライト画面
・ビデオ画面

2.1 画面遷移

これらの画面がどのように重ね合わされるかを以下に示します。

●グラフィック1画面モード



※ :SCREEN@ で変更。

□ :アンダースキャンモード画面

(通常のコンピュータの画面のように画面内の四角の枠内にのみ表示されます。)

□ :オーバースキャンモード画面

(テレビの画面のように画面の枠いっぱいに表示されます。)

●グラフィック２画面モード

SCREEN命令でグラフィック画面を２画面にすることができます。

グラフィック画面
16色
テキストまたは
スプライト画面
※
→
←
256色
※
→
←
32768色
テキストまたは
スプライト画面
※
→
←
グラフィック
画面
ビデオ画面
32768色
スプライト
画面
SIMPOSE
ON/OFF
SCREEN1/0
SCREEN1/0

グラフィック画面
16色
16色
0
1
※
32768色
16色
0
1
グラフィック
画面
ビデオ画面
SIMPOSE
ON/OFF
↑↓※
（注１）
↑↓※
16色
32768色
0
1
※
32768色
32768色
0
1

□ :アンダースキャンモード画面

（通常のコンピュータの画面のように画面内の四角の枠内にのみ表示されます。）

□ :オーバースキャンモード画面

（テレビの画面のように画面の枠いっぱいに表示されます。）

※：SCREEN@ 命令で変更。グラフィック２画面モードの場合は、変更するページを SCREEN1 命令でアクティブにしてから変更します。



16色モード画面が表示されている場合は、スーパーインポーズはできません。SIMPOSE命令の前に、16色モード画面を非表示にしておきます。

2.2 画面モード

●グラフィック1画面(SCREEN 0)の場合

各画面モードにおける画面の大きさ(ドット密度)と表示色は次のとおりです。

画面モード		16色モード	32768色モード	256色モード
グラフィック画面	画面の大きさ	640×480(ドット)	320×240(ドット) 512×256(ドット)	640×480(ドット)
	表示色	4,096色中の16色	32,768色	1,677万色中の256色
テキスト画面	画面の大きさ	80字×25行　または　80字×20行		
	表示色	文字色8色とその反転高輝度表示		
スプライト画面	画面の大きさ	256×256(ドット)	256×240(ドット)	−
	表示色	1つのスプライトで、32,768色中の16色、または32,768色中の256色。		−

画面上の位置は、それぞれの画面の持つ座標で示されます。

グラフィック画面―オリジナルスクリーン座標

テキスト画面―――キャラクタ座標

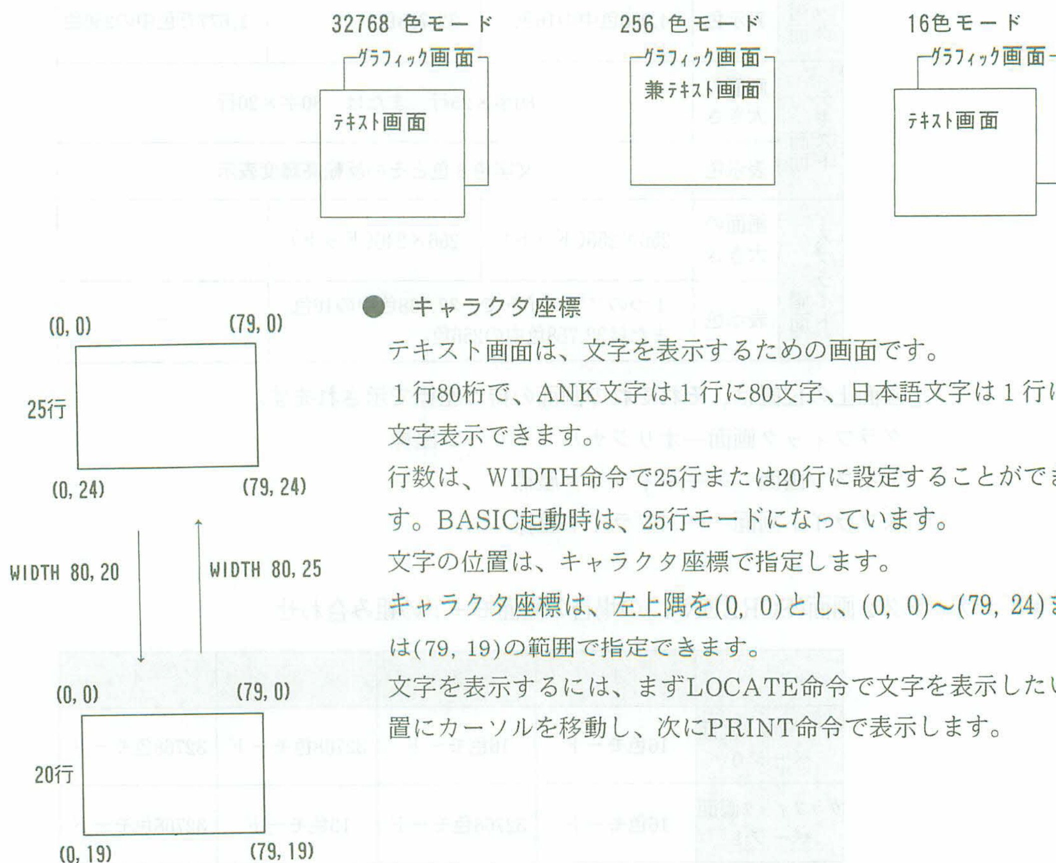
スプライト画面――スプライト座標

●グラフィック2画面(SCREEN 1)の場合の画面モードの組み合わせ

	1	2	3	4
グラフィック画面ページ0	16色モード	16色モード	32768色モード	32768色モード
グラフィック画面ページ1	16色モード	32768色モード	16色モード	32768色モード
テキスト画面	—			
スプライト画面	—			

2.3 テキスト画面

32768色モードと16色モードの場合には、テキスト画面はグラフィック画面と独立しています。つまり、グラフィック画面をスクロールしたりパレットの色の指定を変えても、テキスト画面の文字は影響されません。しかし、256色モードのときは、テキストはグラフィック画面に表示されるため、そのスクロールやパレット切り替えの影響を受けます。



●キャラクタ座標

テキスト画面は、文字を表示するための画面です。

1行80桁で、ANK文字は1行に80文字、日本語文字は1行に40文字表示できます。

行数は、WIDTH命令で25行または20行に設定することができます。BASIC起動時は、25行モードになっています。

文字の位置は、キャラクタ座標で指定します。

キャラクタ座標は、左上隅を(0, 0)とし、(0, 0)～(79, 24)または(79, 19)の範囲で指定できます。

文字を表示するには、まずLOCATE命令で文字を表示したい位置にカーソルを移動し、次にPRINT命令で表示します。

● 文字の色指定

テキスト画面に表示する文字色(文字の表示色)と背景色(文字以外の部分の色)をCOLOR命令で指定します。

文字色は、0～15のコードで指定します。

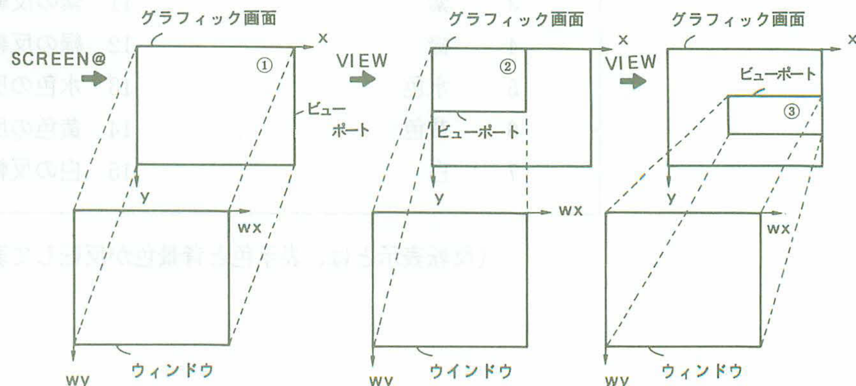
色コード	表示色	色コード	表示色
0	黒	8	黒の反転表示
1	青	9	青の反転表示
2	赤	10	赤の反転表示
3	紫	11	紫の反転表示
4	緑	12	緑の反転表示
5	水色	13	水色の反転表示
6	黄色	14	黄色の反転表示
7	白	15	白の反転表示

(反転表示とは、表示色と背景色が反転して表示されることです。)

2.4 グラフィック画面

● グラフィック座標

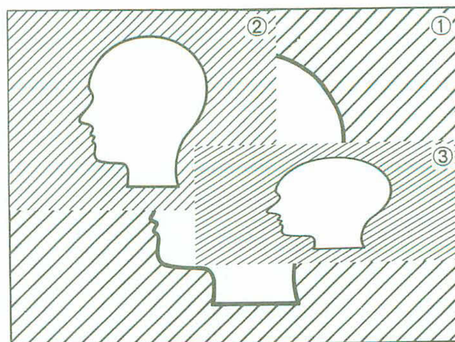
多くのグラフィック命令は、グラフィック画面に直接絵を描くのではなく、別に想定した仮想画面に描きます。この仮想画面をウィンドウと呼び、ウィンドウ上の座標を「ワールド座標」と呼びます。これに対してグラフィック画面の座標を「オリジナルスクリーン座標」と呼びます。ウィンドウ上の座標は、WINDOW命令で定義します。



2つのグラフィック座標は、なにも指定しなければ、一致しています。しかし、グラフィック命令を実行する前に、VIEW命令でウィンドウをグラフィック画面のどこに，どのような大きさで表示するかを設定することによって、画面の一部だけを、グラフィック命令で書き換えることができます。

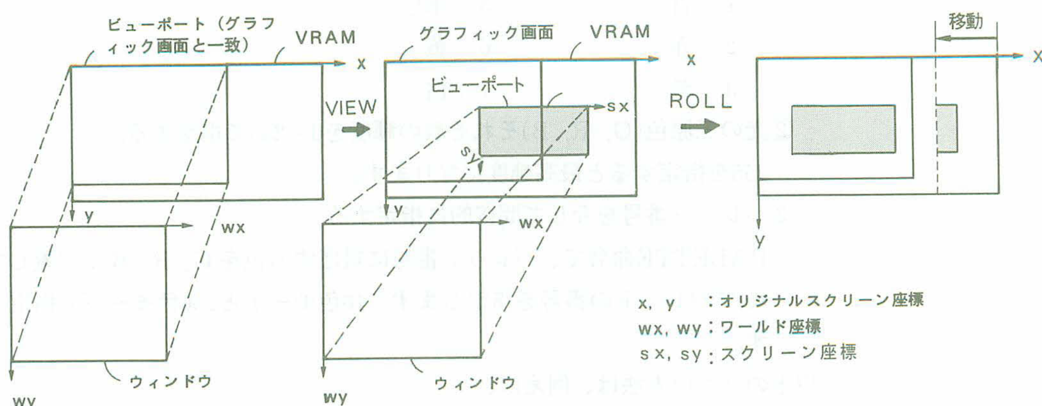
ウィンドウを表示するグラフィック画面上の領域をビューポートと呼びます。

ビューポートの設定を①→②→③と変えて、同じプログラムルーチンを実行しました。

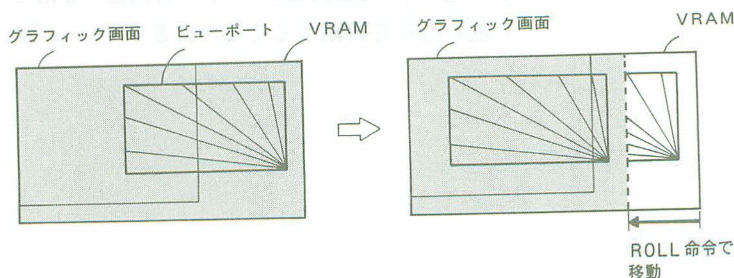


2

ビューポートの大きさを変えたり、ウィンドウとの対応を逆にすることによって、グラフィック画面上に、拡大縮小した絵や向きが逆になった絵を描くことができます。
グラフィック画面の絵は、VRAMと呼ばれるメモリに記憶されています。VRAMはグラフィック画面より大きく、VRAMの左上部分に、常にディスプレイ画面に表示される領域(グラフィック画面)があります。ビューポートは、VRAMの任意の場所に設定することができます。ビューポートの左上隅を(0, 0)とする座標を「スクリーン座標」と呼びます。
ビューポートがグラフィック画面の外に設定された場合、グラフィック画面の外に描いた絵は、ROLL命令を使ってグラフィック画面へ移動させることができます。



例えば、直線を使って描いた絵が、下の図のようにグラフィック画面からはみ出しているとき、ROLL命令を使ってVRAMの内容全体を移動しグラフィック画面に表示することができます。



ワールド座標上の位置は、最終参照座標(LP)からの相対位置によって指定することもできます。

LPは直前のグラフィック命令で指定した座標です。

ただし、WINDOW命令、VIEW命令実行後は、LPはこれらの命令で最初に指定した点の座標値になります。

● 色の指定

色の指定には、3つの方法があります。

①色番号0～7で指定する。

すべての画面モードで8色のみ指定できます。

0：黒または透明色	4：緑
1：青	5：水色
2：赤	6：黄
3：紫	7：白

②光の三原色(G, R, B)それぞれの輝度を0～255で指定する。

255を指定すると最高輝度になります。

③パレット番号を介して間接的に指定する。

PALETTE命令で、パレット番号に対応する色をG, R, Bで定義しておき、%nでパレットの番号を指定します。16色モードと256色モードで指定できます。

以上の3つの方法は、例えば、

```
LINE (60,40)-(260,200),PSET,1,BF
LINE (60,40)-(260,200),PSET,[160,255,0],BF
LINE (60,40)-(260,200),PSET,%1,BF
```

のように指定します。

その他に、COLOR命令で、<前景色>を指定しておくと、各グラフィック命令で色の指定を省略することができます。

⑴ 色番号0～7で指定する方法

次の８色が表示されます。

色番号	表示色	色番号	表示色
0	黒または透明色	4	緑
1	青	5	水色
2	赤	6	黄色
3	紫	7	白

⑵ 光の３原色(緑、赤、青)それぞれの輝度を0～255で指定する方法

この指定方法では、画面モードにより次の数の色が表示できます。

16色モードでは　8色

256色モードでは　256色

32768色モードでは32768色

［g，r，b［,i］］で指定します。

●＜g＞，＜r＞，＜b＞は、それぞれ緑、赤、青の輝度を示し、それぞれ0～255の範囲の整数で指定します。
指定した数値の該当する段階に応じて実際の輝度がとられ、表示色が決定されます。

●＜i＞は32768色モードのときのみ意味を持ち、０か１を指定するかまたは省略します。＜i＞に１を指定すると、SIMPOSE ONの状態のときにその部分は＜g＞，＜r＞，＜b＞の指定にかかわらず透明色となり、後ろのビデオ画面が表示されます。

●16色モードでは、指定値は２段階のどちらかに分けられます。
＜g＞，＜r＞，＜b＞の３色それぞれについて２段階の輝度がとられ、８色(2×2×2=8)が表示できます。

●256色モードでは、指定値によって＜g＞と＜r＞は８段階、＜b＞は４段階の輝度がとられ、
256色(8×8×4=256)が表示できます。

●32768色モードでは、指定値によって32段階の輝度がとられ、
32768色(32×32×32=32768)が表示できます。

256色モードでは、指定値は256段階の輝度のどれかに分けられますので、16,777,216色(256×256×256)が指定できることになります。

パレットの初期設定は次のようになっています。

16色モード

パレット番号	[G, R, B]	パレット番号	[G, R, B]
0	[0, 0, 0]	8	[64, 64, 64]
1	[0, 0, 128]	9	[0, 0, 255]
2	[0, 128, 0]	10	[0, 255, 0]
3	[0, 128, 128]	11	[0, 255, 255]
4	[128, 0, 0]	12	[255, 0, 0]
5	[128, 0, 128]	13	[255, 0, 255]
6	[128, 128, 0]	14	[255, 255, 0]
7	[128, 128, 128]	15	[255, 255, 255]

256色モード

パレット番号	[G, R, B]
0	[0, 0, 0]
1	[0, 0, 127]
2	[0, 0, 191]
3	[0, 0, 255]
4	[0, 63, 0]
5	[0, 63, 127]
6	[0, 63, 191]
⋮	⋮
254	[255, 255, 191]
255	[128, 255, 255]

```
P=0
FOR G1=0 TO 255 STEP 32
 FOR R1=0 TO 255 STEP 32
  FOR B1=0 TO 255 STEP 64
   G=G1:R=R1:B=B1
   IF G1<>0 THEN G=G+31
   IF R1<>0 THEN R=R+31
   IF B1<>0 THEN B=B+63
   PALETTE P, [G, R, B ]
   P=P+1
  NEXT B1
 NEXT R1
NEXT G1
```



PALETTE命令では、ハードウェアパレットの切り替えが行われるので、同じ画面ですでに表示されていた色も影響を受けます。

また、PALETTE命令実行後の画面では、(1)(2)の指定方法で正しい色が表示されなくなります。

(1)(2)による指定と(3)による指定を混在させないようにしてください。

16色モード
<g>, <r>, <b>

輝度	指定値
1	0 127
2	128 255

256色モード
<g> , <r>

輝度	指定値
1	0～31
2	32～63
3	64～95
4	96～127
5	128～159
6	160～191
7	192～223
8	224～255

<b>

輝度	指定値
1	0～63
2	64～127
3	128～191
4	192～255

32768色モード
<g>, <r>, <b>

輝度	指定値	輝度	指定値	輝度	指定値	輝度	指定値
1	0～7	9	64～71	17	128～135	25	192～199
2	8～15	10	72～79	18	136～143	26	200～207
3	16～23	11	80～87	19	144～151	27	208～215
4	24～31	12	88～95	20	152～159	28	216～223
5	32～39	13	96～103	21	160～167	29	224～231
6	40～47	14	104～111	22	168～175	30	232～239
7	48～55	15	112～119	23	176～183	31	240～247
8	56～63	16	120～127	24	184～191	32	248～255

(3) パレット番号で間接的に指定する方法

PALETTE　n , [g , r , b]でパレット番号 n の表す色を定義しておき、%nでパレット番号を指定します。

n には次の範囲の整数を指定することができます。

● 16色モードのときn＝0～15(4096色中の16色を指定できます。)

● 256色モードのときn＝0～255(16,777,216色中の256色を指定できます。)

< g > , < r > , < b >の意味は、(2)の場合と同様ですが、指定した数値の解釈のしかたは異なります。

16色のモードでは、指定値は16段階の輝度のどれかに分けられますので、4096色(16×16×16)が指定できることになります。

● 画像データ

FMTOWNSで標準のイメージデータファイル(TIFF形式 拡張子.TIF)およびJPEG(Joint Photographic Experts Group)形式ファイル(JFIF形式 拡張子.JPG)の画面表示およびファイルへの保存ができます。

(1)画面表示

画面への表示はLOAD@命令で行います。

拡張子が.TIFのファイルはそのファイルを保存した時と同じ画面モードの時のみ表示できます。

拡張子が.JPGのファイルは、32768色モードの時のみ表示できます。

拡張子が.JPGのファイルを表示するときは、CLEAR命令でDLL領域を確保する必要があります。

例)

```
10 CLEAR,,,,,300*1024
20 SCREEN@ 1
30 LOAD@"×××.JPG"
```

詳しくは、LOAD@命令を参照してください。

(2)ファイルへの保存

イメージデータのファイルへの保存は、SAVE@命令で行います。

拡張子が.TIFのファイルは[圧縮情報]を0または1に設定できます。1に設定すると、LZW圧縮形式で保存されます。LZW圧縮形式で保存されたファイルは画質の劣化がありません。

拡張子が.JPGのファイルは[圧縮情報]を必ず2に設定します。2に設定すると、後ろに圧縮の程度を決めるパラメータが設定できます。各パラメータの値は大きくなればなるほど圧縮率が上がり、保存されるファイルのサイズは小さくなりますが、画質が劣化します。

拡張子が.JPGのファイルを保存するときは、CLEAR命令でDLL領域を確保する必要があります。

例）

拡張子.TIFのファイルを表示した後、JPEG形式でファイルに保存します。

```
10 CLEAR,,,,,300*1024
20 SCREEN@ 1
30 LOAD@"×××.TIF"
40 SAVE@"○○○.JPG",(0, 0)-(319, 239), 0, 2, 2, 25, 25
```

詳しくは、SAVE@命令を参照してください。

● グラフィック画面への文字描画

グラフィック画面に文字を描画するときにはSYMBOL命令を使います。
SYMBOL命令で文字を描画する前に、DEF FONT命令でフォント名を指定しておくと、いろいろな書体を使って描画することができます。（CD-ROMに入っている明朝体、ゴシック体、まる文字等が使えます。）
ベクトルフォントカード（オプション）を装着している場合は、ベクトルフォントも指定することができます。
DEF FONT命令を使う時は、CLEAR命令でDLL領域を確保する必要があります。

例）

```
10 CLEAR,,,,,300*1024
20 DEF FONT "まるもじ　24ドット"
30 SYMBOL(176, 200),"まるもじだぁ〜〜！", 2, 2, 1
```

詳しくは、DEF FONT命令、SYMBOL命令を参照してください。

● グラフィック命令の論理操作

グラフィックの命令のうち、LINE, CIRCLE, PSET命令などでは、指定した色で絵や図形を描くだけでなく、既に画面に表示されている色と指定した色との論理演算をドットごとに行って、その結果の色で描画させることができます。
この機能は、論理操作を記述できる命令で使うことができます。
論理操作は、次のいずれかを指定します。

<論理操作>	機　　能
PSET	(既に表示されている色とは無関係に)指定した色で描画します。
PRESET	(既に表示されている色とは無関係に)背景色で描画します。
AND OR XOR	指定した色と既に表示されている色との論理演算(AND, OR, XOR)を行い、その結果の色で描画します。
NOT	(既に表示されている色とは無関係に)指定した色のビット表現を反転した色で描画します。
MATTE	PUT@Aでのみ指定することができます。 MATTEを指定する場合は、かならず透過色を指定しなければなりません。 PUT@Aのパラメタである〈透明色〉のドットは描画しません。 それ以外の色のドットはPSETと同様に指定色で描画します。
OPAQUE	色コード0または〔0, 0, 0〕のドットを黒でなく背景色で描画します。 それ以外のドットはPSETと同様に指定色で描画します。 ラインスタイルとともに指定すると、線の欠ける部分を背景色で描画します。
PASTEL	既に表示されている色と混色して描画します。混色の比率はPASTEL命令で指定することができます。16色モードでは指定できません。

2

● パターンの指定

(1)タイルストリング

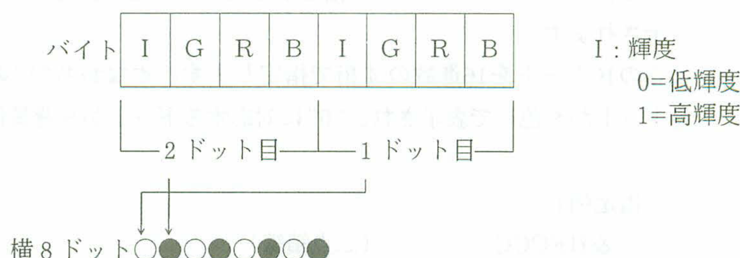
PAINT命令、LINE命令、CIRCLE命令などで、塗りつぶすときのタイルパターンは＜タイルストリング＞で指定します。

以下に16色モードのタイルパターンの指定方法を説明します。

16色モードのタイルパターンは、横８ドット×縦１～32ドットを単位として、指定します。

１ドットを４ビットで表現するので、横８ドットは４バイトを必要とします。

１バイト中のビットの構成は、次のとおりです。



例えば、16進数で、&HA0と指定すると、2ドット目が赤の高輝度になります。

＜タイルストリング＞の指定は、16進数または文字列で指定します。

例）　横８ドット×縦４ドット

○○○●●○○○　－－赤

○○●○○●○○　－－赤

○●○○○○●○　－－黄

●○○○○○○●　－－黄

● 16進数指定の場合

&H00A00A00000AA000E000000E0E0000E0

1列目　2列目　3列目　4列目

● 文字列指定の場合

CHR$(&H00)+CHR$(&HA0)+CHR$(&H0A)+CHR$(&H00)
CHR$(&H00)+CHR$(&H0A)+CHR$(&HA0)+CHR$(&H00)
CHR$(&HE0)+CHR$(&H00)+CHR$(&H00)+CHR$(&H0E)
CHR$(&H0E)+CHR$(&H00)+CHR$(&H00)+CHR$(&HE0)

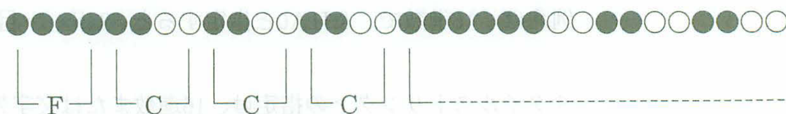
(2)ラインスタイル

LINE命令、CIRCLE命令、CONNECT命令などで、線を描くときの線種は、<ラインスタイル>で指定します。
以下に線種の指定方法を説明します。

線は、<ラインスタイル>で指定する16ドットを単位として、その繰り返しで表示されます。
この16ドットを16進数の4桁で指定します。すなわち"1"のビットに対応するドットが<色>で表示され、"0"に対応するドットが<背景色>で表示されます。

[指定例]

&HFCCC　　　　(二点鎖線)



2.5 スプライト画面

画面モードが32768色モードと16色モードの場合には、ゲームキャラクタなどを動かすためのスプライト画面を表示することができます。スプライト画面が表示された状態では、テキスト画面が表示されないためPRINT文やINPUT文が実行できなくなります。

スプライト画面の大きさは、モードによって異なります。

16色モード



SPRITE ON →



スプライト画面は，グラフィック画面内の任意の位置に重ねることができます。SPRITE SCREENで指定します。

32 768色モード



SPRITE ON →



横方向のドット数が、グラフィックの3/4

↓ SPRITE SCREEN 1



スプライト画面を横方向に拡大して，グラフィック画面と同じ大きさにすることができます。

グラフィック画面のどの位置にスプライト画面を重ね合わせるかは、SPRITE SCREEN命令で指定します。

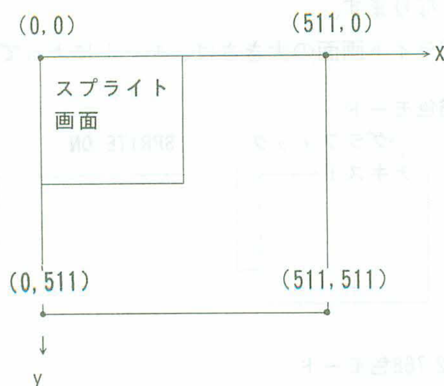
スプライト画面の一部がグラフィック画面からはみ出すような位置を指定することはできません。

□：アンダースキャン、画面内の四角の枠内にのみ表示され、枠の外が黒く残る。

□：オーバースキャン、テレビの画面のように全画面に表示する。

● スプライト座標

スプライトは、グラフィック座標とは別の、独自の座標(スプライト座標)で、位置指定します。座標の範囲はスプライト画面より大きく、画面外にあるスプライトは、移動命令により、スプライト画面上に移動させることができます。



ただし、スプライトを表示できるのは次の範囲です。

16色モードのとき	(0, 2)〜(255, 255)
32768色モードのとき	(0, 2)〜(255, 239)

2

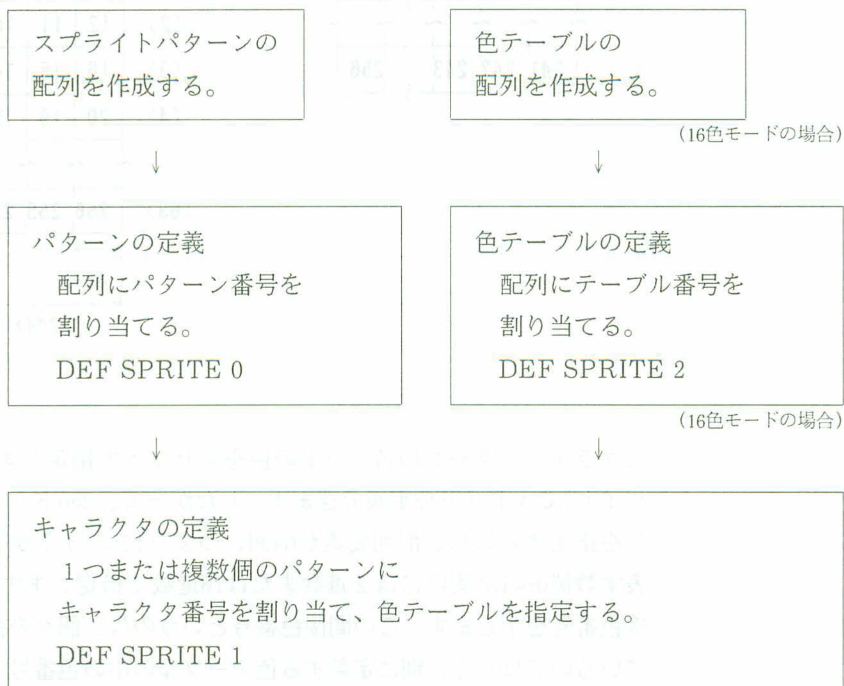
● スプライトキャラクタの作成

スプライトキャラクタの最小単位は、16×16ドットのスプライトパターンで、これを1つまたは複数個組み合わせて、スプライトキャラクタを作成します。

最初にスプライトパターンの配列を作成し、これにパターン番号を割り当てます。

配列の指定方法に、16色モードと32768色モードの、2つのモードがあります。

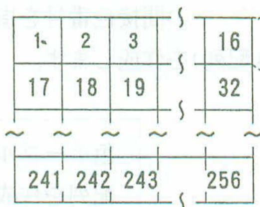
16色モードでは、パターンの各ドットの間接色番号を指定し、別にその番号に対応する色を定義した色テーブル(配列)を作成します。



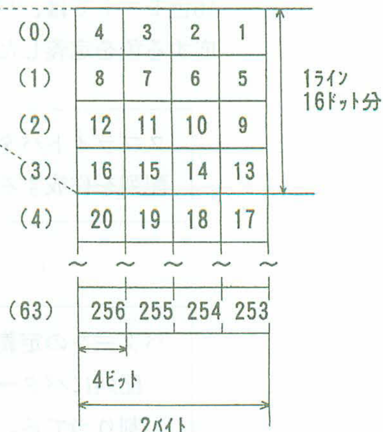
同一の画面に16色モードと32786色モードのスプライトを混在させて使用できます。ただし一つのスプライトキャラクタには、同じモードのパターンしか組み合わせることができません。

(1) 16色モードのスプライトパターン

画面上のスプライトパターン
(16 × 16 = 256 ドット)

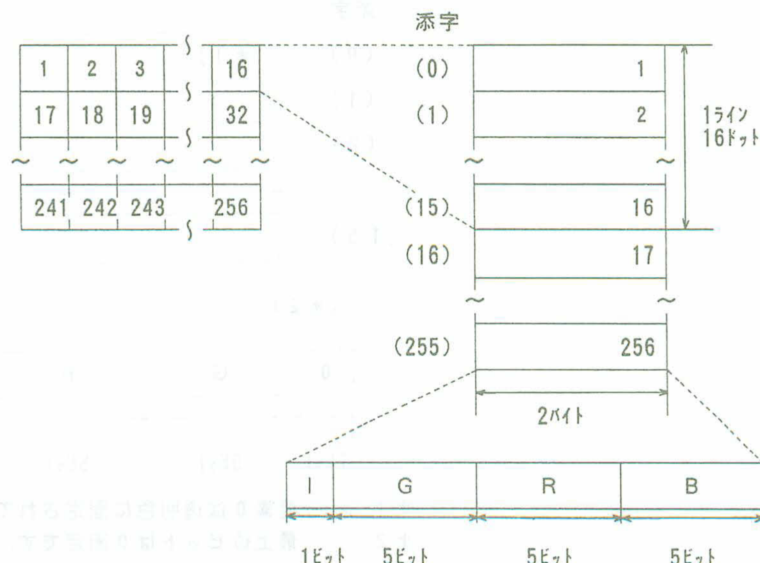


配列指定
DIM 整数配列名(63)
添字



スプライトパターンの各ドットの色を4ビットで指定します。配列の1要素(2バイト)で4ドットを定義できます。したがって、256ドットのスプライトパターンを指定するには、配列要素が64個、つまり128バイトが必要です。4ビットが表す数値0～15(実際には2進数または16進数で指定します)は、そのドットの間接色番号を示します。この間接色番号というのは、個々の番号が固有の色を持っているのではなく、別に定義する色テーブルの中の色番号との対応で、実際の表示色が決まるものです。

(2) 32768色モードのスプライトパターン

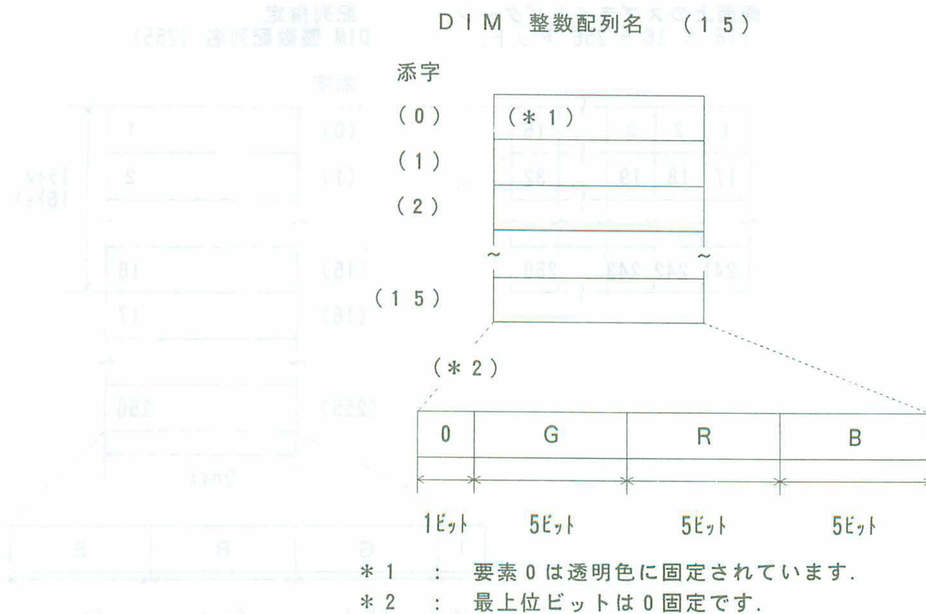


スプライトパターンの各ドットの色を２バイトで指定します。配列の１要素(2バイト)で１ドットを定義します。したがって、256ドットのスプライトパターンを指定するには、配列要素が256個、つまり512バイトが必要です。各ドットを指定する２バイトは、上図のとおり、G(緑)、R(赤)、B(青)の輝度を表しており、各ドットの表示色を直接指定します。配列式は次のとおりです。

配列名(添字) $= -\mathrm{I} \times 2^{15} + \mathrm{G} \times 2^{10} + \mathrm{R} \times 2^{5} + \mathrm{B}$

- Ｉは１か０を指定します。０を指定すると、そのドットは不透明となり、１を指定すると、透明となります。
- G, R, Bはそれぞれ0〜31の値を指定します。０が最低、31が最高輝度です。

● 色テーブル



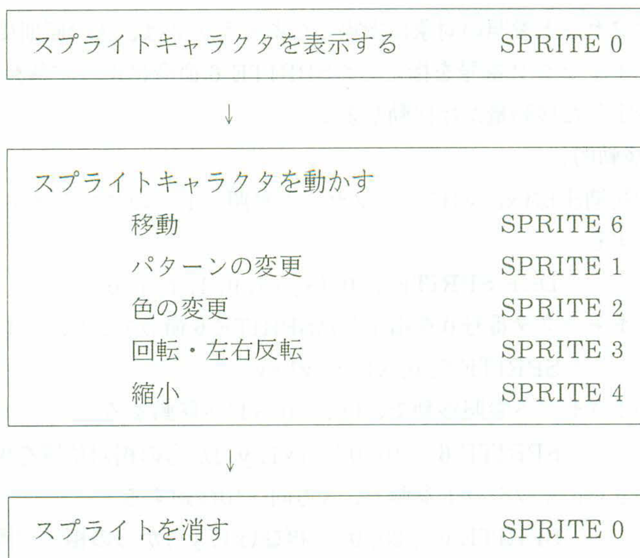
スプライトパターンの定義で指定した間接色番号(0～15)の実際の表示色を、対応する添字番号の要素で指定します。1個の間接色番号に対応する実際の表示色を、1要素(2バイト)で指定します。1つの色テーブルは、16要素(32バイト)から成ります。各要素は次のように指定します。

配列名(添字)$=G\times2^{10}+R\times2^{5}+B$

G, R, Bはそれぞれ緑、赤、青の輝度で、0～31の整数です。

● スプライトキャラクタの表示・移動

スプライトキャラクタを移動するには、スプライト座標で移動先を指定します。



- スプライトキャラクタの状態を調べるには、SPRITE関数を使います。
- 複数のパターンからなるキャラクタを移動する場合には、直前に、SPRITE TIME命令を実行して、パターンが乱れないようにする必要があります。
- スプライトの回転は、16×16ドットのパターンの単位で行われます。複数個のパターンを組み合わせたキャラクタでは、個々のパターンがその場で回転するため、キャラクタ全体が回転した状態にはなりません。

● オフセット参照キャラクタの移動

キャラクタ番号を省略したSPRITE 6 命令を実行すると、オフセット参照の対象に指定したキャラクタ全部が同時に移動します。

オフセット参照の対象に指定したキャラクタは、次の原則に従って移動します。

「キャラクタ番号を指定したSPRITE 6 命令によって移動した最後の位置から、指定した移動量だけ移動します。」

［移動例］

①初期座標(x, y)に、オフセット参照＝1 のスプライトキャラクタ0を定義する。

```
DEF SPRITE 1, 0, (x, y), 0, 1, 1, 1, 0
```

②キャラクタ番号0を指定したSPRITE 6 命令で、(x1, y1)へ移動する。

```
SPRITE 6, 0, x1−x, y1−y
```

③オフセット参照移動で、(x1+10, y1)へ移動する。

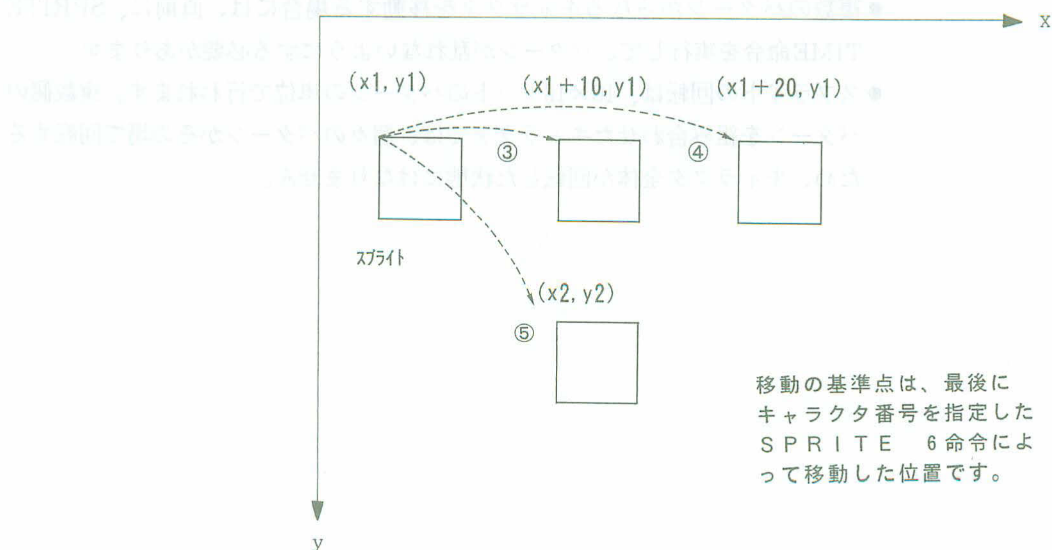
```
SPRITE 6, , 10, 0 ' (x1, y1)からの相対位置を指定
```

④さらにオフセット参照で、X方向へ10移動する。

```
SPRITE 6, , 20, 0 ' 再び(x1, y1)からの相対位置を指定
```

⑤キャラクタ番号0を指定したSPRITE 6 命令で、(x2, y2)へ移動する。

```
SPRITE 6, 0, x2−x1, y2−y1 ' (x1, y1)からの相対位置を指定
```

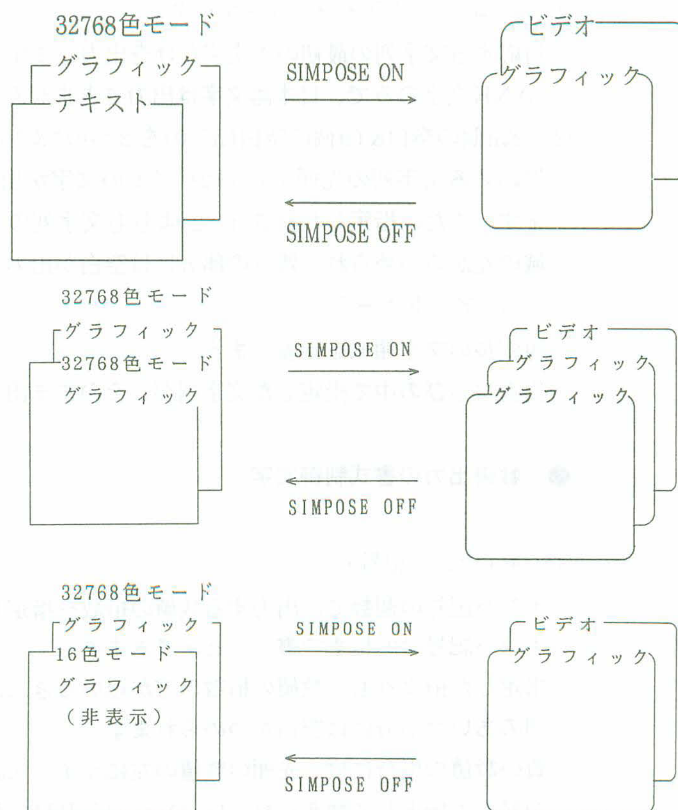


2

2.6 ビデオ画面

16色モード画面が表示されている場合は、スーパーインポーズモードにできません。したがって、グラフィック1画面モードのときは、画面モードが32768色モードの場合にのみ、ビデオ画面を表示することができます。ビデオ画面が表示された状態では、テキスト画面が表示されないため、PRINT文やINPUT文が実行できなくなり、代わりにビデオ映像とグラフィック画面が合成されて表示(スーパーインポーズ)されます。

グラフィック2画面モードの場合は、SIMPOSE命令の前に、SCREEN命令で16色モード画面を非表示にしておきます。





ビデオ画面を表示するには、オプションのビデオカード(FMT-411)が必要です。

3. 印刷

書式制御文字で、印刷形式（書式）を指定することができます。

3.1 書式制御文字

LPRINT USING命令では、書式文字列で、印刷形式を指定します。書式文字列で使用する制御文字には、次のものがあります。これらの制御文字は、PRINT USINGで、画面出力の書式を指定するときにも使用します。

● 文字出力の書式制御文字

(1) !（エクスクラメーションマーク）

対応する文字列の最初の1文字だけを出力します。
（ANK文字のみで、日本語文字は出力できません。）

(2) &n個の空白&(n個の空白(n≧0)を2つの“&”で囲む)

対応する文字列の先頭からn+2バイトの文字が出力され､残りの文字は無視されます。また、指定した長さ(n+2)よりも文字列の方が短いときは、文字は文字領域の左からつめられ、残りの部分には空白が出力されます。

(3) @（アットマーク）

可変長の文字領域を定義します。
出力ならびの中で指定した文字列が、そのまま出力されます。

● 数値出力の書式制御文字

(1) #（ナンバ記号）

ナンバ記号の個数で、出力する数値の桁数を指定します。
ナンバ記号は24桁まで書くことができます。
指定した桁よりも、数値の桁数の方が短いときは、数値は右づめで出力され、左側のあいた部分には空白がつめられます。
負の数値の場合には、先頭の数値の左にマイナス記号が出力されます。マイナス記号も1桁として数えられ、1つのナンバ記号に対応します。



実際の数値が指定された桁数で表せないときは、数値の先頭にパーセント記号（%）が出力されます。
数値の丸めの結果、指定桁数を超えた場合でも、丸められた数値の前にパーセント記号（%）が出力されます。

(2) .（小数点）

数値領域の任意の個所に小数点を代入することを指示します。

小数点に続いてナンバ記号がある場合は、小数点以下の桁数がナンバ記号の個数と同じだけ出力されます。

(3) +（プラス）

書式制御文字の最初または最後にプラス記号を書くと、数値の前または後にその数値の符号（プラスまたはマイナス）がつけられます。

(4) −（マイナス）

書式制御文字の最後にマイナス記号を書くと、数値が負の場合に、数値の後にマイナス符号がつけられます。

(5) **(2個のアスタリスク)

数値の左にある空白部分と同じ桁数だけアスタリスクが出力されます。

書式制御文字の先頭で指示します。

#（ナンバ記号）の機能も兼ねます。すなわち2個のアスタリスクは、2桁分の領域を確保します。

(6) ￥￥(2個の円記号)

数値の直前の空白位置に円記号が1つ出力されます。

書式制御文字の先頭で指示します。

#（ナンバ記号）の機能も兼ねます。すなわち2個の円記号は2桁分の領域を確保します。

(7) **￥(2個のアスタリスクと円記号)

(5)と(6)の両方の機能を果たします。

書式制御文字の先頭で指示します。

**￥は3桁分の領域を確保します。

(8)　，（コンマ）

書式制御文字“.”（小数点）のすぐ左側に置きます。

整数部分が右側から３桁ごとに、コンマで区切られます。

コンマも１桁として数えられ、1個のコンマに１桁が使用されます。

(9)　^^^^(4個の矢印）と^^^^^(5個の矢印）

桁指定文字のナンバ記号の後に置きます。

指数形式で出力され、4個の矢印はＥ±nnが出力される領域が確保され、5個の矢印はＤ±nnnが出力される領域が確保されます。

#(ナンバ記号）の数だけ有効数字の桁数がとられ、指数はそれに合わせて調整されます。

(10)　_（アンダスコア記号）

アンダスコア記号に続く１文字を、書式制御の意味を持たない文字としてそのまま出力します。

アンダスコア記号自身を文字として出力するためには、アンダスコア記号を２つ並べます。

●　定数文字列の出力

<書式文字列>の中に、書式制御文字以外の文字を含むとき、これら定数文字列は書式制御文字に従って変換された文字の前後に、そのまま表示されます。

4. データ入力

データ入力するときに使用する命令です。データを入力するには、データをプログラム内に持つか、キーボードから入力するか、あるいはファイルから入力します。

4.1 プログラム内のデータ

プログラムの中でデータを定義して使うときには、DATA命令とREAD命令を使います。

- データ（定数）を定義します。－－－－－－－DATA　1, 2, 3, 4, ・・・
- 定数を変数に読み込みます。－－－－－－－－READ　A, B, C, D, ・・・

READ命令は、定義された定数と変数を、1対1に対応させて読み込みます。
複数のDATA命令やREAD命令で定義された定数や変数も、連続して定義された一連のものとして扱われます。
したがって次のプログラムは、すべて同じ結果になります。

例1
```
10 READ X, Y, Z
20 DATA 1, 2, 3
```

例2
```
10 READ X
20 READ Y
30 READ Z
40 DATA 1, 2, 3
```

例3
```
10 READ X, Y
20 READ Z
30 DATA 1
40 DATA 2, 3
```

読み込むデータの開始行番号をRESTORE命令で指定することができます。

例4
```
10 RESTORE 50        'X=3, Y=4, Z=5
20 READ X, Y, Z      の順で読み込まれます。
30 DATA 1
40 DATA 2
50 DATA 3
60 DATA 4, 5, 6
```

4.2 キー入力

キー入力を利用するための命令や関数には、次のようなものがあります。

(1)INKEY$関数

キーが押されていなければ、空文字列を返します。次のプログラムは空文字列ならループし、何かのキーが押されると次へ進みます。

"PUSH ANY KEY"などのメッセージで処理を始めるときなどに使う方法です。

```
1000 A$=INKEY$
1010 IF A$="" THEN 1000
1100 PRINT "次処理開始"
```

(2)INPUT$関数

指定した文字数が入力されるまで待ちます。

A$=INPUT$(n)　　　nは文字数

カーソルも、入力した文字も表示されません。

(3)INPUT命令

メッセージとカーソルを表示し、入力状態になります。

入力された1個以上のデータを1個以上の変数に代入します。

(4)LINE　INPUT命令

メッセージとカーソルを表示し、入力状態になります。

入力された1行全体が、1つの変数に代入されます。

5. データファイル

ファイルには、シーケンシャルファイルとランダムファイルの2つの種類があります。

シーケンシャルファイル：ファイルの先頭から順にデータを書き込み、読み出すときも、先頭から順に読み出します。

ランダムファイル　　　：一定の長さのデータ（レコード）を単位として、先頭から何番目かを指定して書き込み・読み出しを行います。

データファイルの保存と読み込みの前には、OPEN命令、後にはCLOSE命令が必要です。

オープンされたフロッピィディスクを、クローズせずに取り換えてしまうと、新しく入れたフロッピィディスクの内容が壊されることがあります。

5.1 ランダムファイルの入出力

ランダムファイルの指定は、ディスク上のファイルに対してだけ可能です。

①ファイルのオープン

ファイルのモードをR（ランダムファイル）に指定し、ファイル番号を割り当てます。以後、このファイル番号でファイルを指定します。

```
OPEN "R", #1, "A:DFILE1"
```

②バッファの定義

ランダムファイルバッファをフィールドに分割し、各フィールドに文字変数を割り当てます。文字変数以外の変数は使用できません。

```
FIELD #1, 10 AS A$, 10 AS B$, ・・
```

③データの代入

文字変数へ値を代入します。LSETは左詰め、RSETは右詰めで代入します。

```
LSET A$="FUJITSU":LSET B$="TOWNS"
```

- 数値は、変換関数MKI$/MKL$/MKS$/MKD$を使って、文字形式に変換し、代入します。

```
INPUT A&' 数値をキーから入力
LSET A$=MKL$(A&)' 文字に変換して代入
```

④書き込み

PUT #1, 3' 3番目のレコードに書き込む

読み出し

GET #1, 3' 3番目のレコードから読み出す

● 読み出したデータを数値に戻すには、変換関数CVI/CVL/CVS/CVDを使います。

A=CVS(A$)' A$に読み込んだデータを変換

⑤クローズ

CLOSE #1

5.2 シーケンシャルファイルの入出力

①ファイルのオープン

ファイルのモードをI（入力)、O（新規ファイルに出力)、A（追加出力）のどれかに指定し、ファイル番号を割り当てます。

以後、このファイル番号でファイルを指定します。

OPEN "O", #1, "A:DFILE2"

②書き込み

WRITE#1, A$, B$

式の結果をコンマ（,）で区切って出力します。

● 読み込みの場合

OPEN命令で、入力モードに指定し、INPUT#命令やLINE INPUT#命令で読み込みます。

INPUT#1, A$, B$

データを読み込み、変数に代入します。INPUT#命令では、コンマ（,)、コロン（:)、またはCR、LFなどが読み込まれると、データの区切りとみなされます。

LINE INPUT#命令では、CR、LFのみがデータの区切りとみなされます。

③クローズ

CLOSE #1

6. マウス／パッド

2

6.1 マウスを使うプログラム

マウスを使うときには、必ずMOUSE　0　で始まり、MOUSE　5　で終わります。

● **マウスが右クリックされるのを待つ**

右クリックされるのを待って処理を進めるには、以下のようにします。

① マウス機能を初期化します。－－－ MOUSE　0

② マウスカーソルを(x0, y0)に
表示します。－－－－－－－－－－－ MOUSE　1, x0, y0, 1

③ 右クリックされるまでループ
して待ちます。 { WHILE　MOUSE(2, 1)=0
WEND

④ マウスカーソルを消して、
次の処理に進みます。－－－－－－－ MOUSE　5

⑤ 次の処理　－－－－－－－－－－－－ PRINT　"次処理開始"

● **マウスによる位置指定**

マウスボタンを押した位置と離した位置を読み取ることができます。
押した点と離した点を結ぶ線を対角線とする四角形を描く手順を次に示します。

① マウス機能を初期化します。－－－ MOUSE　0

② マウスカーソルを(x0, y0)
に表示します。－－－－－－－－－－ MOUSE　1, x0, y0, 1

③ 左ボタンを押すまで、ループします。WHILE　MOUSE(2, 0)=0
WEND

④ 押した位置を代入します。－－－－ X1=MOUSE(4, 0)
Y1=MOUSE(5, 0)

⑤ 左ボタンを放すまでループします。 WHILE MOUSE(2, 0)<0
WEND

⑥ 離した位置を代入します。－－－－ X2=MOUSE(7, 0)
Y2=MOUSE(8, 0)

⑦ マウスカーソルを消します。－－－ MOUSE　5

⑧ 指定位置に四角を描きます。－－－ LINE　(X1,Y1)-(X2,Y2),PSET,2, B

● マウスカーソルの形状の設定

マウスカーソルは、16×16ドットまたは32×32ドットで表示されます。
初期化された状態では、マウスカーソルは、左上向きの矢印をしています。
16×16ドットマウスカーソルの形状を設定するには、MOUSE 2命令を用いて、andパターンとドットパターンの２つのパターンを32バイトの文字列で指定します。

MOUSE 2, andパターン, ドットパターン

マウスカーソルの形状は、＜ドットパターン＞と＜andパターン＞を重ね合わせ、各ビットの論理演算の結果で表示されます。
つまり、<andパターン>が「０」であるドットに対し、<ドットパターン>も「０」のときは黒に、「１」のときは白にします。
また、<andパターン>が「１」であるドットに対しては、<ドットパターン>が「０」のときは地の色に、「１」のときは白にします。
これをまとめると次の表になります。

<table>
<tr><td colspan="2" rowspan="2">パターン</td><td colspan="2">and</td></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td rowspan="2">ドット</td><td>1</td><td>白</td><td>白</td></tr>
<tr><td>0</td><td>地</td><td>黒</td></tr>
</table>

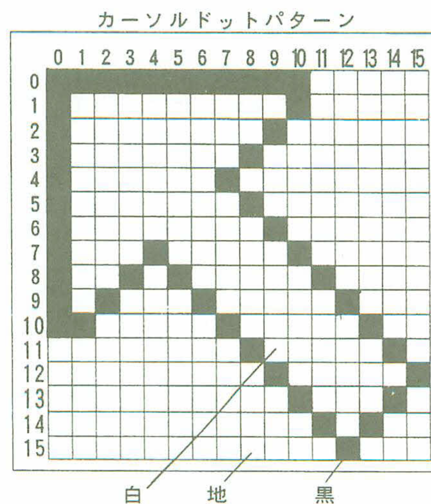
初期化された状態でのマウスカーソルのドットパターンは、下図のようになっています。

＜andパターン＞

																データ	
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	00	1F
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	00	1F
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	00	3F
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	00	7F
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	00	FF
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	00	7F
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	00	3F
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	00	1F
0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	1	08	0F
0	0	0	1	1	1	0	0	0	0	0	0	0	1	1	1	00	07
0	0	1	1	1	1	1	0	0	0	0	0	0	0	1	1	3E	03
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	FF	01
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	FF	80
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	FF	C1
1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	FF	E3
1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	FF	F7

＜ドットパターン＞

																データ	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00	00
0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	7F	C0
0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	7F	80
0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	7F	00
0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	7E	00
0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	7F	00
0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	7F	80
0	1	1	1	0	1	1	1	1	1	0	0	0	0	0	0	77	C0
0	1	1	0	0	0	1	1	1	1	1	0	0	0	0	0	63	E0
0	1	0	0	0	0	0	1	1	1	1	1	0	0	0	0	41	F0
0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	00	F8
0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	00	7C
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	00	3E
0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	00	IC
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	00	08
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00	00



32×32ドットのマウスカーソルの形状を設定するには、MOUSE 6の命令を用いて、andパターンとドットパターンの２つのパターンを配列で指定します。

MOUSE 6，モード，andパターン，ドットパターン

マウスカーソルの形状は、16×16ドットマウスカーソルと同じようにandパターンとドットパターンを重ね合わせ、各ビットの論理演算の結果で表示されます。

MOUSE 6命令では、パターンの登録を配列によって行うことで､GET@、GET@A命令を利用してMOUSE 2より簡単にパターンを作成できます。

次ページにそのプログラム例を示します。

例）

単色マウスカーソルの形状設定プログラム

```
10 SCREEN @0:COLOR 7, 0:CLS              ' 16色モード  前景色=7, 背景色=0
20 DIM PAND%(64), PDOT%(64)              ' ANDパターン64×2(128), ドッ
                                           トパターン64×2(128)
30 MOUSE 0                               ' マウスの初期化を行う
40 LINE(0, 0)-(31, 31), PSET, 2, B       ' 2(赤色)で□を書く
50 GET@(0, 0)-(31, 31), PAND%, 0         ' ANDパターン作成 0 色は透過色
                                           で取り込む
60 GET@(0, 0)-(31, 31), PDOT%, 2         ' DOTパターン作成 2 (赤色)は描
                                           画色で取り込む
70 MOUSE 6, 0, PAND%, PDOT%              ' マウスカーソル形状設定
80 MOUSE 1, 100, 100, 1                  ' 白□のマウスカーソルを100, 100
                                           の位置に表示
90 WHILE MOUSE(2, 0)=0 :WEND             ' 左ボタンが押されるまで待つ
100 END                                  ' 終了
```

カラーマウスカーソルの形状設定プログラム

```
10 SCREEN @0:COLOR 7, 0:CLS              ' 16色モード  前景色=7, 背景色=0
20 DIM PAND%(64),PDOT%(256)              ' ANDパターン, DOTパターン
30 MOUSE 0                               ' マウスの初期化を行う
40 LINE(0, 0)-(31, 31), PSET, 2, B       ' 2(赤色)で□を書く
50 GET@(0, 0)-(31, 31), PAND%, 0         ' ANDパターン作成 0 色は透過色
                                           で取り込む
60 GET@(0, 0)-(31, 31), PDOT%            ' DOTパターン作成 2 (赤色)は描
                                           画色で取り込む
70 MOUSE 6, 1, PAND%, PDOT%              ' マウスカーソル形状設定
80 MOUSE 1, 100, 100, 1                  ' 赤□のマウスカーソルを100, 100
                                           の位置に表示
90 WHILE MOUSE(2, 0)=0 :WEND             ' 左ボタンが押されるまで待つ
100 END                                  ' 終了
```

7. 文字列処理

文字の種類や文字列の長さを返したり、文字の変換をする命令です。

ランダムファイルへ出力するときは、数値は文字型に変換して出力します。入力のときには、逆に文字列を数値に復元します。

Kで始まる名前と、Kのない名前の命令の対があるものは、文字列の長さの数え方が異なります。

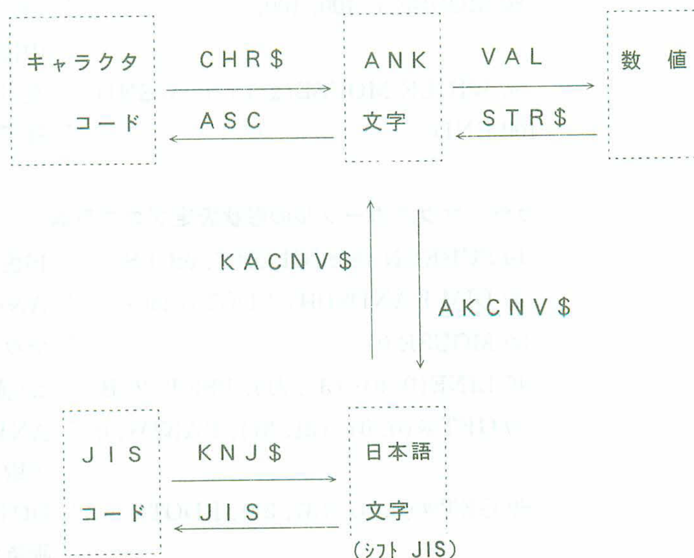
(1)文字列の長さをバイト数で数える。(日本語文字は1文字を2バイトと数える)

LEN/INSTR/LEFT$/MID$など

(2)ANK文字、日本語文字共に1文字と数える。

KLEN/KINSTR/KLEFT$/KMID$など

● 文字コード変換



2

8. 音楽／音声

ブザー	BEEP
音楽演奏の並行動作を制御する	BGM
音楽演奏する	PLAY, PLAY関数, PLAY ON/OF/STOP, PLAY@
パートを割り当てる	PART, PART関数
音色を設定する	VOICE, LOAD@, VOICE SET
音色データを配列にコピーする	VOICE COPY
MIDIポートへデータを送る	OUTM
音色を記録・再生する	PCMREC, PCMPLAY
EUPデータを配列に設定する	LOAD@
サウンドメッセージを再生する	SMSGPLAY

- 音楽・音声・ブザーを取り扱う命令です。
- FMTOWNS本体に内蔵された音源を使って音楽を演奏することができます。
- 内蔵の音色データをBASICプログラムで変更したり、TownsSOUNDで作った音色データや、マイクで録音したサンプリングデータと入れ換えたりして、オリジナルの音を作ることができます。
- MIDIポートに、MIDI楽器を接続して演奏することができます。

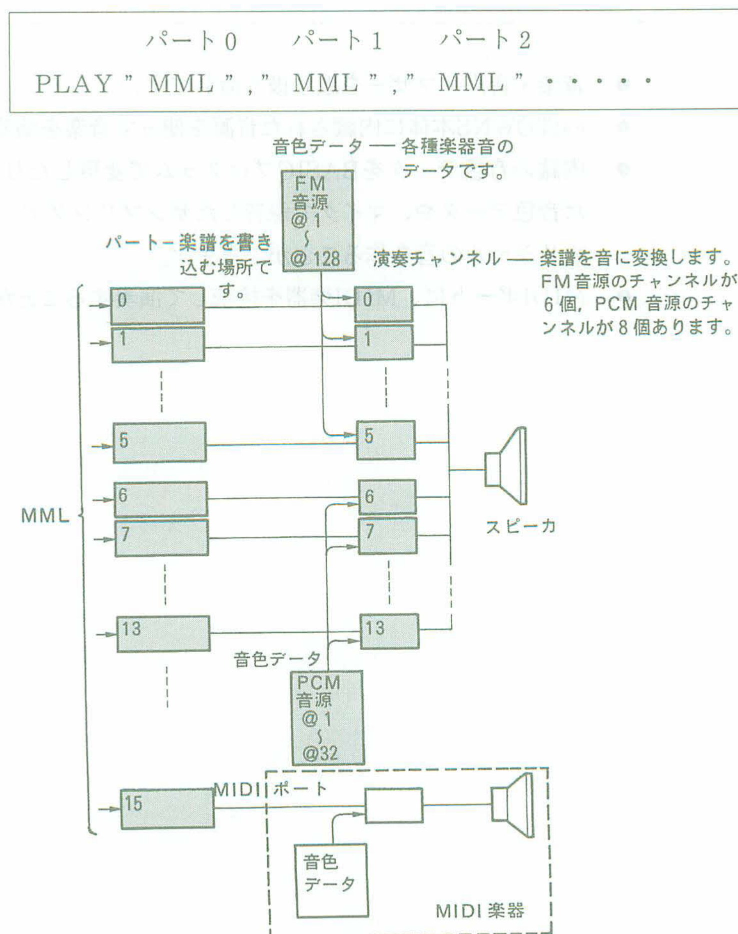
8.1 音楽演奏

PLAY命令で、音楽データ（楽譜）を各パートに送り込みます。そのパートの演奏チャンネル（奏者）は、音楽データで指定された音色データ（楽器）を使って演奏を行います。音楽データ、演奏チャンネル、音色データは、それぞれ楽譜、奏者、楽器に相当します。音楽データは、MML(Music Macro Language)という言語で記述します。

PLAY@命令では、EUPファイルを使って音楽を演奏します。

PLAY命令では、16のパートに音楽データを送ることができますので、FMTOWNS内蔵の14の演奏チャンネルの他にMIDI楽器などの外部の演奏チャンネルを使えば、16和音を出すことができます。

	パート0	パート1	パート2	
PLAY	" MML " ,	" MML " ,	" MML "	・・・・・



8.2 音色データ

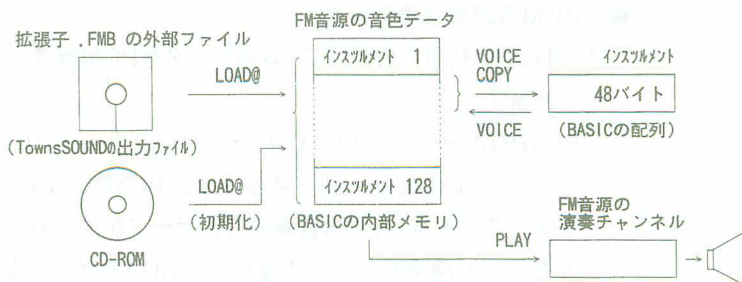
● FM音源

FM音源の音色データは、128個あり、1つの音色データ(インスツルメント)は、48バイトで構成されます.

TownsSOUNDを使って作成した音色データ（～. FMB）を、LOAD@命令で読み込んで使用することができます。

音色データを初期状態に戻したいときには、LOAD@命令でBASICのCD-ROM内にある音色データファイルを読み込みます。

VOICE COPY命令で音色データをBASICの配列に読み出して、その音色を変更することができます。



● PCM音源

PCM音源の音色データは、32のインスツルメントと128のサウンドデータから構成されます。

1つのインスツルメントは、128バイトで構成されます。サウンドデータの長さは一定していません。

PCM音源のインスツルメントは、音域別にいくつかのサウンドデータを指定し、その組み合わせで1つの音色を作ります。

BASICのCD-ROMにある各種の音色データファイルをLOAD@命令で読み込んで使用することができます。

また、TownsSOUNDを使って作成した音色データ(～. PMB）を、LOAD@命令で読み込んで使用することもできます。

FM音源と同様、VOICE COPY命令でインスツルメントをBASICの配列に読

み出して、その音色（インスツルメント）を変更することができます。
音色データを初期状態に戻したいときには、LOAD@命令でBASICのCD-ROMにある音色データファイルを読み込みます。



● PCM音源の音声データ

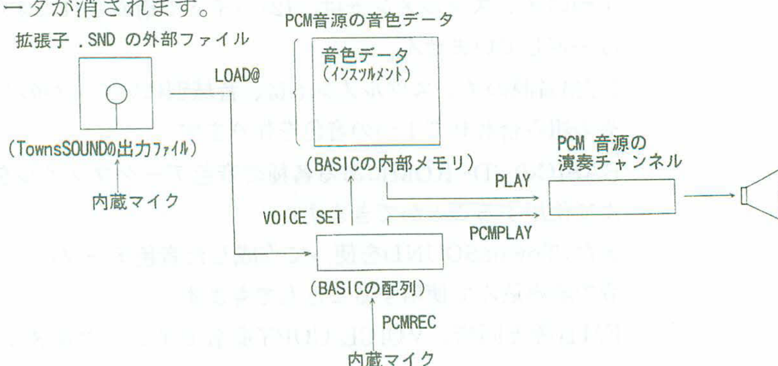
PCM音源の利用方法として、音声データをPCM録音し、再生することができます。

(1) 録音再生

PCMREC命令は、FMTOWNSのマイクから入力した音声を、サンプリングしてBASICの配列に読み込みます。PCMPLAY命令は、この音声を再生します。このとき、PCM音源の音色データが消されますので、この後でPCM音源を使った演奏をしたいときには、LOAD@命令でBASICのCD-ROMにある音色データを読み直す必要があります。

(2) 音楽演奏

PLAY命令を使って、録音した音声データに音階を付けて演奏することができます。このためには、VOICE SET命令で、音声データをPCM音源の音色データとして登録する必要があります。このときにも、PCM音源の他の音色データが消されます。



● EUPデータ

EUPデータとは、アプリケーションソフト「EUPHONY」で作成される標準MIDI形式を基に設計された演奏データ形式です。

データを配列に読み込み、演奏することができます。

LOAD＠命令でEUPデータを配列に読み込みます。

PLAY＠命令でEUPデータを演奏します。

8.3 MML

MMLは、音高、音長、音量、テンポなどを指定する一種の言語です。

MML一覧表

MMLコマンド	機　能（PCM音源 / FM 〃）	初期値
A～G	音　名	——
+(#)	半音上げる(シャープ)	——
−	半音下げる(フラット)	——
Rn	休　符	——
Ln	音　長(1≦n≦384)	4
.(ピリオド)	音長を1.5倍にする	——
On	オクターブ(1≦n≦8)	4
<	1オクターブ下げる	——
>	1オクターブ上げる	——
Nn	音　高	——
Tn	テンポ(30≦n≦280)	120
Vn	音　量(0≦n≦15)	8
Qn	1音中の音長の割合(1≦n≦10)	8
&	前後の音をつなぐ(タイ)	——
{ }n	連符. nの長さの中で{ }内の音符の個数を等分の長さで演奏する	——
@n	n番の音色に変更する(FM音源 1≦n≦128, PCM音源 1≦n≦32)	14
@Vn	音量を細かく調整する(0≦n≦127)	——
( )n	n回くり返す(2≦n≦5)	2
%Ln	左寄りから音を出す(1≦n≦15)	15
%Rn	右寄りから音を出す(1≦n≦15)	15
%C	中央から音を出す	——
%Sn/m	拍子の宣言(n/m拍子) PLAY(1)関数を返す値に影響する	4/4
Un	ピッチ(音程)を微妙にずらす(−127≦n≦127)	——
〔	ボリュウムの値を1つ下げる	——
〕	ボリュウムの値を1つ上げる	——
T±n	テンポを変える(0≦n≦15)	——
!n	アクセント(1≦n≦15) 次の音符のボリュームを指定する	——

MML コマンド	機　能（PCM音源 / FM 〃）	初期値
Hn	MIDI楽器のチャンネル番号を設定する（0≦n≦15，n=255）	——
@Cn/m	コントロール番号nに対応する機能の値をmに設定する（0≦n≦127，0≦m≦127）	——
%On	出力ポートを設定する（MIDI楽器 0≦n≦7，内蔵音源240≦n≦255）	——

● 音名（A, B, C, D, E, F, G）、半音（+, #, −）

A〜GはそれぞれA=ラ、B=シ、C=ド、D=レ、E=ミ、F=ファ、G=ソの音名に対応しています

A〜Gの直後に，半音を表す#、+（シャープ）、−（フラット）を付けることができます。また、後に、整数(1〜384)を付けて音長を設定することができます。

（☞「●　音長」）

● 休符(Rn)

Rの後に、整数(1〜384)を付けて休符の長さを設定することができます。

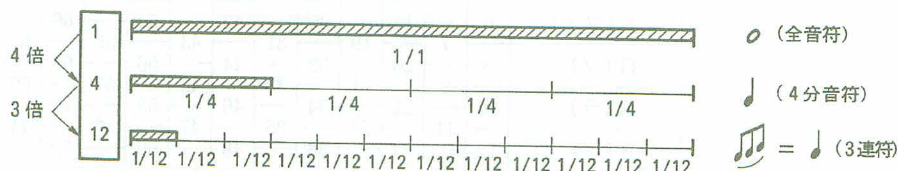
（☞「●　音長」）

● 音長（Ln，または n）

n は1〜384の整数で 1 を全音符として、n の逆数が音の長さを表します。このコマンドから以降は、指定した音長で演奏されます。また音名、半音のコマンドの後に n を追加すると、その音だけその音長で演奏されます。

例　4 分音符のド…………L4CまたはC4
　　16分音符のファの# …L16F#またはF#16またはF+16
　　全音符のソの♭……L1G−またはG−1

音長の決め方の簡単な方法は、n 分音符ならその n が求める値です。ただし、3 連符のときは n は3倍になります。



n	1		2		4		8		16	
記号	𝅝	𝄻	𝅗𝅥	𝄼	♩	𝄽	♪	𝄾	𝅘𝅥𝅯	𝄿
名称	全音符	全休符	2分音符	2分休符	4分音符	4分休符	8分音符	8分休符	16分音符	16分休符

● 音長(ピリオド)

付点音符は、A～Gコマンドの音長の後にピリオドを追加することで表します。付点はその音長の1.5倍の長さになるので"A4."にすると音長4(4分音符)の1.5倍になり音長 $n=\frac{8}{3}$ の長さで演奏されます。

例　付点4分音符のド　♩.…………C4.

　　付点8分音符のファの#　#♪.…F+8.　またはF#8.

● オクターブ(On, <, >)

n は1～8までの整数です。なお、基準周波数440Hzのラの音(A)はオクターブ4(O4)です。

"<"を指定すると、現在演奏されている高さから1オクターブ下がり、">"を指定すると、1オクターブ上がった高さで演奏されます。

● 音高(Nn)

音高は通常A～Gまでのコマンドとオクターブ指定で表わしますが、Nで指定することもできます。n は0～96までの整数で指定します。13がO1C、96がO7Bに対応しています。n はオクターブおよび音名と次のように対応づけられます。

対　応　表

音名 ＼ オクターブ	O0		O1		O2		O3		O4		O5		O6		O7	
C(ド)	1	2	13	14	25	26	37	38	49	50	61	62	73	74	85	86
D(レ)	3	4	15	16	27	28	39	40	51	52	63	64	75	76	87	88
E(ミ)	5		17		29		41		53		65		77		89	
F(ファ)	6	7	18	19	30	31	42	43	54	55	66	67	78	79	90	91
G(ソ)	8	9	20	21	32	33	44	45	56	57	68	69	80	81	92	93
A(ラ)	10	11	22	23	34	35	46	47	58	59	70	71	82	83	94	95
B(シ)	12		24		36		48		60		72		84		96	

例1　O4A+のときのn

　　音名AでO4の所は58ですが、半音上がっているので59が求める値です。

2

例２　O3B−のときのｎ

B−はシの半音下がった音なので、求める値は47です。

● テンポ（Tn）

ｎは30〜280の整数です。ｎは１分間に４分音符をｎ回数える速さです。

● 音量（Vn）

そのパートの音量を指定します。ｎは0〜15の整数で15が最大となります。

また０ではそのパートの音は、出力されません。ただし、FM音源では、最低レベルの音が出ます（☞「● 音量設定」）。

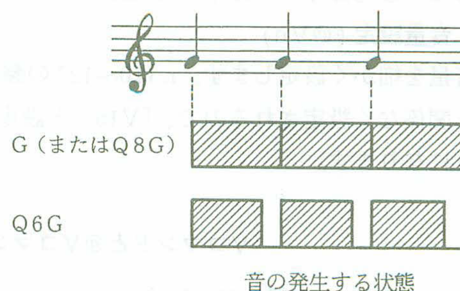
● １音中の音長（Qn）

１音中の音の長さの割り合いを設定します。

同じ音高の音を続けて演奏する場合、音色によっては、第２音以降の音が区別できないことがあります。その場合は、１音中の音長を設定して音を区切ることができます。

n	1	2	3	4	5	6	7	8	9	10
音長／8	1／8	2／8	3／8	4／8	5／8	6／8	7／8	8／8	9／8	10／8

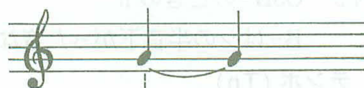
例　音程がソ（G）の４分音符を続けて演奏する場合



● タイ（&）

続けて演奏する音をつなぎます。タイ（&）を指定した次のコマンドが違う音高や記号の場合は、タイ（&）指定したものは“Q８”と同じになります。

例　音名がソ（G）の４分音符をタイ（&）で指定します。



G&G

音の発生する状態

● 連符（｛　｝n）
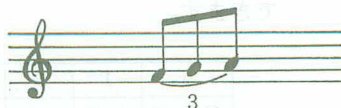
n で指定する音長の中で｛　｝内の音符を繰り返します。

したがって 1 つの音符の長さは $\frac{1}{\text{音符の数}\times n}$ となります。

$\left(\frac{1}{64} \leqq \frac{1}{\text{音符の数}\times n} \leqq 1\right)$

例　8 分音符の 3 連符

{FGA}8



● 音色（@n）

指定した音色番号に切り替えます。初期値は、FM音源では n=14（PIANOの音色）、PCM音源ではドラムの音色になっています。FM音源の音色番号とPCM音源の音色番号は「付録2　音色番号一覧表」のとおりです。

● 音量設定（@Vn）

音量を細かく設定します。n は0～127 の整数です。このコマンドはVnコマンドと関係なく設定されるので、「V15」と設定しても「@V0」と設定すれば音量は0になります。

Vコマンドと@Vコマンドの n の対応

V	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
@V	85	87	90	93	95	98	101	103	106	109	111	114	117	119	122	125



音色により音量のバランスをとってあるために、音色によってVと@Vの対応は異なります。

● くり返し ((　)n)

(　)でくくった中のMMLをn回繰り返して演奏します。nは2〜5が指定でき、省略すると2を指定したものとみなされます。

● 音の位置指定 (%Ln, %Rn, %C)

音を中央から出すか、左寄りに出すか、右寄りに出すかを指定します。nは1〜15で、PCM音源の場合、15のとき最も左（または右）、1のときは中央よりわずかに左（または右）となります。%Cは中央から出します。nを省略すると15を指定したものとみなされます。FM音源の場合は，nを指定してもしなくても15として解釈されます。

● 拍子の宣言 (%Sn/m)

演奏データがn/m拍子であることを宣言します。%Sだけでn/mを略すと4/4と解釈されます。何小節目を演奏中かを返すPLAY関数の値に影響します。

● ディチューン (Un)

ピッチ（音程）の微妙なズレを設定します。nは−127〜＋127の範囲で指定します。

● 相対ボリューム (〔, 〕)

“〔”を指定すると、ボリュームの値が一つ下がり、“〕”を指定すると、ボリュームの値が一つ上がります。

● 相対テンポ (T±n)

nは0〜15の整数で、現在演奏されているテンポを変えます。

変更後のテンポは、30〜280の範囲になければなりません。

● アクセント (!n)

nは0〜15の整数で、次に来る音符だけを指定のボリュームにします。

●チャンネル番号 (Hn)

MIDI楽器のチャンネル番号を設定します。nは0〜15の整数です。また、nに255を指定するとOMNI受信になります。

●音源のコントロール (@Cn/m)

nで指定したコントロール番号に対応する機能の値をmに設定します。nおよびmは0〜127の整数です。コントロール番号の7と10は内蔵音源に使用します。

7	ボリューム	m=127のとき、最大になります。
10	パンポット	m=127のとき、最も右になります。

●出力ポート (%On)

出力ポートを設定します。0〜7の整数でMIDI楽器の、240〜255の整数で内蔵音源の出力ポートを指定します。

9. 動画再生

ムービーファイルを開く	MOVIE OPEN
ムービーファイルを再生する	MOVIE PLAY
ムービーファイルを閉じる	MOVIE CLOSE
オープンしているファイルの細かい情報を知る	MOVIE INFO
再生の仕方を設定する	DEF MOVIE

● BASICでは、2種類のムービーファイルを再生するLiveMotion機能を持っています。

・LiveMovie……動画ファイル(.MVE)を再生します。

・LiveAnimation……アニメーションファイル(.MMM)を再生します。

● この機能はDLL機構を使用しているので、CLEAR命令でDLL領域を確保する必要があります。

●動画ファイル

動画ファイル(.MVE)は、ビデオソースを取り込んで作る動画です。LiveMovieまたはTownsGEAR内のムービーキャプチャーで作成できます。

動画ファイルの再生では、音の再生を基準にして絵を表示します。そのため、CPUやハードディスク等の性能によっては絵がスキップ(コマ落ち)され動きが荒くなりますが、総再生時間はどのマシンでも変わりません。またこの方式はデータの一部を読み出しながら再生を行うので、メモリは少なめでも動作しますが、ハードディスクや光磁気ディスク等、高速・大容量の外部記憶装置が必要です。

動画ファイルの再生に必要なDLL領域の大きさは、次の通りです。

CD−ROMから再生する場合…………約400KB〜600KB
ハードディスクから再生する場合……約300KB〜400KB

●アニメーションファイル

アニメーションファイル(.MMM)は、キャストと呼ばれる部品群の表示で作る動画です。LiveAnimationで作成できます。

アニメーションファイルの再生では、テンポ(コマとコマの間の時間)という時間要素はありますが、キャストの動きをすべて再生するので、コマ落ち等は発生しません。ただし、すべてのキャストをメモリ上に読み込んでから再生するので、データによっては大量のメモリが必要になりますし、ファイルをオープンしてから再生するまでに少し時間がかかることがあります。

アニメーションファイルは圧縮されたキャストを展開するので、再生に必要なDLL領域の大きさはファイルによって異なります。目安としては、再生するファイルサイズの5倍以上です。

例)

```
10 CLEAR ,,,,,400*1024
20 MOVIE OPEN "SAMPLE.MVE"
30 MOVIE PLAY
40 MOVIE CLOSE
50 END
```

詳しくは、MOVIE OPEN/PLAY/CLOSE命令やCLEAR命令を参照してください。

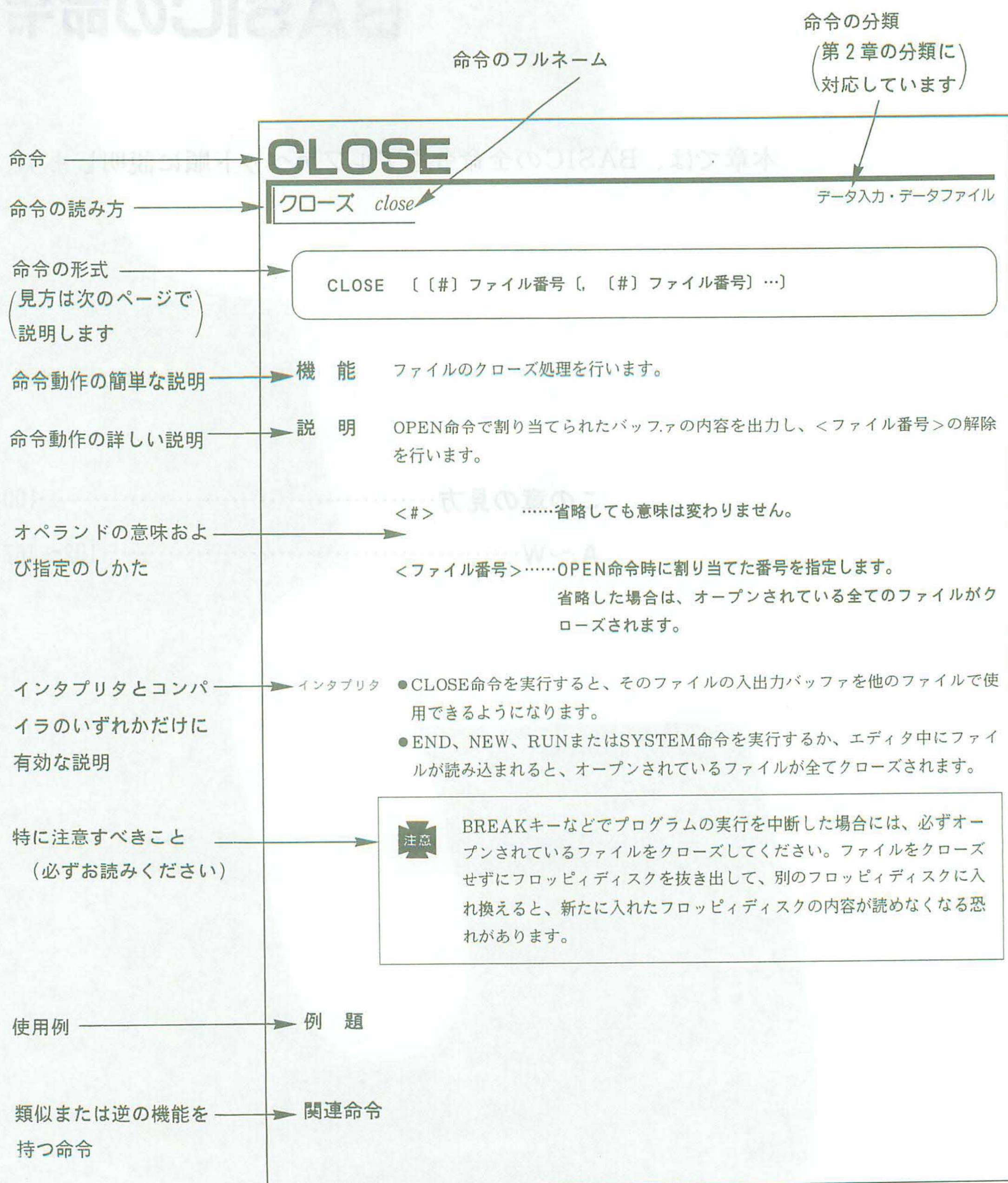
第3章

BASICの命令

本章では、BASICの全命令をアルファベット順に説明します。

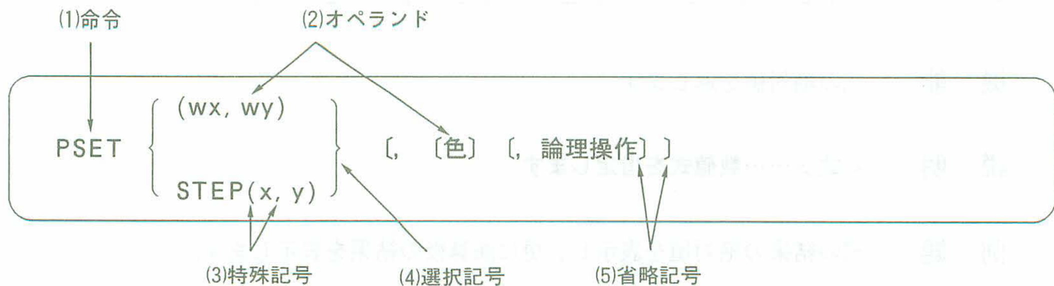
この章の見方

各命令の説明は、次のような構成になっています。



命令の形式の見方

命令の形式は、命令の記述方法を示しています。



(1) **命令または関数**

英大文字の部分は、命令または関数です。
ANK文字でそのままのつづりを記述します。

(2) **オペランド**

英小文字、数字または日本語の部分は、オペランドです。
この命令を実行するときの対象となる変数や定数を記述します。

(3) **特殊記号**

以下の特殊記号は、そのまま記述します。

( )	小かっこ	—	ハイフン
,	コンマ	=	等価記号
:	コロン	”	ダブルクォーテーション
;	セミコロン		

(4) **選択記号** { }

複数の項目からいずれか一つを選んで記述します。

(5) **省略記号** 〔 〕

省略可能な項目を示しています。
省略したときの動作については、その命令の「説明」を、お読みください。

(6) **繰り返し記号** (…)

一行の許される範囲で、その項目を繰り返し記述することができます。

ABS

アブソルート *absolute* 計算

ABS（式）

機　能　式の絶対値を返します。

説　明　<式>……数値式を指定します。

例　題　式の結果の絶対値を表示し、更に演算後の結果を表示します。

```
1000 A=10:B=-20
1100 PRINT ABS(A)
1200 PRINT ABS(B)
1300 PRINT A+B
1400 PRINT ABS(A)+ABS(B)
1500 END

結果： 10
       20
      -10
       30
```

AKCNV$関数

エー・ケー・コンバート・ダラー関数　*ANK to kanji convert$*　文字列処理

AKCNV$（文字式）

機　能　文字列中のANK文字を、対応する日本語文字に変換します。

説　明　ANK文字の英数字、カタカナ、記号は、各々日本語文字の英数字、カタカナまたは記号に変換されます。変換できない文字はそのまま残ります。

<文字式>……ANK文字を含む文字列を指定します。
文字列の長さは、日本語文字に変換した後の長さが255バイト以内とします。

●変換後の長さが255バイトを超えた場合は、次のエラーメッセージが表示されます。



ERROR15　文字列の長さが許される範囲を超えています

例　題　ANK文字を対応する日本語文字列に変換して、ANK文字と日本語文字を表示します。

```
1000 A$="FUJITSU"
1100 B$=AKCNV$(A$)
1200 PRINT A$
1300 PRINT B$
1400 END

結果：FUJITSU
      Ｆ Ｕ Ｊ Ｉ Ｔ Ｓ Ｕ
```

関連命令　KACNV$関数：AKCNV$とは逆に、日本語文字をANK文字に変換します。

ASC関数

アスキー関数　*ascii*　　　　文字列処理

ASC（文字式）

機　能　ANK文字をキャラクタコードに変換します。

説　明　<文字式>で指定した文字列の、最初の文字のキャラクターコード（数値）を返します。

<文字式>……文字列を指定します。
　　　　　　文字式に空文字列を指定してはいけません。

●空文字列を指定した場合は、次のエラーメッセージが表示されます。



ERROR5　関数または命令文の使い方が正しくありません

例　題　文字列の最初の文字"C"をキャラクタコード67に変換して表示します。

```
1000 A$="CHARACTER CODE"
1100 A=ASC(A$)            ' 先頭文字'C' のコードを変数A に与える
1200 PRINT A
1300 END

結果：67
```

関連命令　CHR$関数：ASC関数とは逆に、キャラクタコードを文字に変換します。
JIS関数　：日本語文字をJISコードに変換します。

ATN関数

アークタンジェント関数 *arc tangent*

ATN（式）

機　能　三角関数アークタンジェント（逆正接）の値を返します。

説　明　<式>……数値式を指定します。

●ATNの演算結果は、角度（単位：ラジアン）を表し、$-\pi/2$から$\pi/2$の範囲の数値になります。

ラジアン＝π×角度（°）／180（°）

●式の型が、倍精度の場合は倍精度の、単精度の場合は単精度の結果を返します。

例　題　入力した倍精度型の数値の、三角関数アークタンジェント値を求めて表示します。

```
1000 INPUT A
1100 B=ATN(3.14159!/180*A)              ' 単精度の値をB に与える
1200 PRINT B
1300 PRINT
1400 INPUT A#
1500 B#=ATN(3.14159265359#/180*A#)      ' 倍精度の値をB#に与える
1600 PRINT B#
1700 END

結果：? 5.6
      9.74289E-02

      ? 0.6987521065
      1.21949203507821D-002
```

関連命令　TAN関数：三角関数タンジェント（正接）の値を返します。

BEEP

ビープ *beep*　　音楽/音声

```
BEEP [ { 0 / 1 } ]
```

機　能　ブザーを鳴らします。

説　明　<0><1>……ブザーを鳴らすか、止めるかを指定します。

0：ブザーを止めます。

1：BEEP　0か、またはBEEPだけの命令が実行されるまで、ブザーを鳴らします。

省略すると、ブザーを一定時間鳴らします。

例　題　FOR～NEXTループで、音の長さを変えながらブザーを鳴らします。

```
1000  FOR I=1 TO 100
1100    BEEP  1              ' ブザーを鳴らす
1200    FOR J=1 TO I: NEXT J
1300    BEEP 0               ' ブザーを止める
1400    FOR K=I TO 100: NEXT K
1500  NEXT I
1600  END
```

関連命令　PLAY:音楽を演奏します.

BGM

ビー・ジー・エム　*back ground music*　　音楽/音声


```
BGM { 0 }
    { 1 }
```

機　能　PLAY命令による音楽演奏の並列動作をコントロールします。

説　明　並列動作とは、PLAY命令で音楽を演奏しながら、他の命令の実行を行うことをいいます。

<0><1>……並列動作をするかしないかを指定します。
　0 ：並列動作を行いません。
　1 ：並列動作を行います。

- 起動時には、1が設定されています。
- 0（並列動作を行わない）を指定した場合は、PLAY命令の実行が終了するまで、次の命令は実行されません。

例　題　音楽を演奏しながら円を描きます。次に、演奏終了後に円を描きます。

```
1000 CLS
1100 BGM 1
1200 PLAY "@20L405CDEFEDCREFGAGFER"
1300 CIRCLE (320,250),50,4,,.6,.45,F
1400 CIRCLE (350,205),10,7,,,,F
1500 CIRCLE (365,220),10,7,,,,F
1600 CIRCLE (350,205),2,0,,,,F
1700 CIRCLE (365,220),2,0,,,,F
1800 BGM 0
1900 PLAY "@68V1503BDCR"
2000 CIRCLE (450,280),20,4,,.6,.45,F
2100 CIRCLE (500,285),15,4,,.6,.45,F
2200 END
```

関連命令　PLAY　　:音楽を演奏します。
PLAY関数:PLAY文の実行中かどうかの状態を返します。

CALLM

コール・エム　*call machine program*

機械語プログラム

1 CALLM　エントリオフセット [, 引数] …

機　能　引数付きで、プロシジャを実行します。

説　明　メモリ中に読み込まれているプロシジャを引数付きで呼び出して実行します。

<エントリオフセット>……プロシジャの実行開始アドレスを、プロシジャ領域の先頭からのオフセットによって指定します。
ロング型整数で指定します。

<引数>　……プロシジャへの入出力用引数として、整数またはロング型整数の数値定数、数値変数、配列の要素などを指定します。
文字型定数、文字型変数、配列の全要素を指定したい場合は、VARPTR命令を使用し、アドレスを引数として渡します。アドレスはロング型整数で指定します。

●入力用引数としてアドレスを指定した場合、そのアドレスの内容をプロシジャが変更することによって、出力用引数としても使用できます。

●引数は、プロシジャへの入出力用引数として、値やアドレスを指定します。式を指定することはできません。

インタプリタ　●インタプリタでは、引数は16個まで指定できます。

関連命令　VARPTR関数：変数アドレスまたはシステムアドレスを返します。

C

2 戻り値=CALLM（エントリオフセット［, 引数］…）

機　能　引数付きでプロシジャを実行し、戻り値を返します。

説　明　メモリ中に読み込まれているプロシジャを、引数付で呼び出して実行させます。

<戻り値>……エントリオフセットで指定されるプロシジャの戻り値として、ロング型整数値を返します。

<エントリオフセット>……プロシジャの実行開始アドレスをプロシジャ領域の先頭からのオフセットによって指定します。ロング型整数で指定します。

<引数>　……プロシジャへの入力用引数として、数値定数、数値変数、配列の要素などを指定します。
ロング型整数で指定します。
文字型定数、文字型変数、配列の全要素を指定したい場合は、VARPTR文を使用し、アドレスを引数として渡します。

●引数は、プロシジャへの入出力用引数として、値やアドレスを指定します。
式を指定することは使用できません。
●入出力引数としてアドレスが渡された場合、そのアドレスの内容を変更することによって、出力用引数としても使用できます。

インタプリタ　●インタプリタでは、引数は16個まで指定出来ます。

CDBL

コンバート・ダブル *convert to double*

計算

```
CDBL (式)
```

機 能 整数値、ロング型整数値または単精度実数値を倍精度実数値に変換します。

説 明 <式>……整数値、ロング型整数値または単精度実数値を数値式で指定します。

●型変換するだけで、精度そのものは変わりません。

例 題 入力された整数値を、倍精度実数値に型変換し表示します。

```
1000 INPUT A%
1100 A#=CDBL(A%)
1200 PRINT A#
1300 INPUT B
1400 B#=CDBL(B)
1500 PRINT B#
1600 END

結果：? 12
      12
      ? 7894562
      7894562
```

関連命令 CINT：ロング型整数値、単精度実数値または倍精度実数値を整数値に変換します。
CLNG：整数値、単精度実数値または倍精度実数値をロング型整数値に変換します。
CSNG：整数値、ロング型整数値または倍精度実数値を単精度実数値に変換します。

CDINF

シーディ・インフォメーション *CD information*

```
CDINF 配列名
```

機　能　CDの内容を調べます。

説　明　現在のCDの曲数や時間を調べます。

<配列名>……返される情報がはいる配列を指定します。
配列は、5つの要素からなり、各要素には、以下の情報が返されます。

配列の添字1：CDの内容を返します。
　1：音楽用
　2：プログラム用
　3：音楽、プログラム共用
2：総演奏時間を分で返します。
3：総演奏時間を秒で返します。
4：総演奏時間をフレームで返します。
5：曲数を返します。

- 配列は整数配列のみ有効です。添字0および6以上の要素の値は不定です。
- DIM命令で宣言していない配列を使用すると、要素数11個（添字が0～10）の配列が自動的にとられます。

関連命令　CDSTAT　：CDドライブの状態を返します。
CDSTIME$：CD中の各曲の開始時間を返します。

CD PAUSE/CONT/STOP

シーディ・ポーズ/コント/ストップ　*CD pause/continue/stop*　CD演奏

```
CD PAUSE
CD CONT
CD STOP
```

機　能　CDの演奏を一時停止、再開、中止します。

説　明
- ●CD　PAUSE　　CDの演奏を一時停止します。
- ●CD　CONT　　CD　PAUSEで一時停止した演奏を再開します。
- ●CD　STOP　　CDの演奏を中止します。

関連命令　CD　PLAY：CDを演奏します。

CD PLAY

シーディ・プレイ　*CD play*　　CD演奏


```
            ┌┌ 開始トラック番号〔, 終了トラック番号〕              ┐┐
CD  PLAY    ││ (開始分, 開始秒, フレーム〔, トラック〕)             ││
            └└ 〔−(終了分, 終了秒, フレーム〔, トラック〕)〕        ┘┘
```

機　能　CDを演奏します。

説　明　CDの指定した範囲の曲を演奏します。

＜開始トラック番号＞……演奏を開始する曲の位置を、トラック番号で指定します。

＜終了トラック番号＞……演奏を終了する曲の位置を、トラック番号で指定します。

＜開始分, 開始秒, フレーム〔, トラック〕＞……演奏を開始する位置を、CDまたはトラックの先頭からの時間で指定します。

＜終了分, 終了秒, フレーム〔, トラック〕＞……演奏を終了する位置を、CDまたはトラックの先頭からの時間で指定します。

- ＜開始トラック番号＞のみ指定すると、その指定した曲のみを演奏します。
- ＜開始分, 開始秒, フレーム〔, トラック〕＞のみ指定すると、開始位置から最後まで演奏します。
- オペランドを全て省略すると、CDの先頭から全曲を演奏します。
- フレームは秒の下の単位で、75フレームが1秒に相当します。0〜74の整数を指定します。

関連命令　CD　STOP　：CDの演奏を中止します。
CD　PAUSE：CDの演奏を一時停止します。
CD　CONT　：CDの演奏を再開します。

CDSTAT

シーディ・ステイタス　*CD・status*

CD演奏

```
CDSTAT　配列名
```

機　能　CDドライブの状態を返します。

説　明　現在のCD−ROMドライブの状態（演奏中のトラック番号、演奏時間など）を返します。

<配列名>……返される情報が入る配列を指定します。
配列は、8つの要素からなり、各要素には、以下の情報が返されます。

配列の添字1：状態を、0（演奏していない）または1（演奏中である）で返します。
2：演奏時間を分で返します。
3：演奏時間を秒で返します。
4：演奏時間をフレームで返します。
5：演奏しているトラック番号で返します。
6：曲内演奏時間を分で返します。
7：曲内演奏時間を秒で返します。
8：曲内演奏時間をフレームで返します。

- ●配列は整数配列のみ有効です。添字0および9以上の要素の値は不定です。
- ●DIM命令で宣言していない配列を使用すると、要素数11個（添字0～10）の配列が自動的にとられます。

関連命令　CDINF　　：CDの内容を調べます。
CDSTIME$：CD中の各曲の開始時間を返します。

CDSTIME$

シーディ・タイムダラー　*CD time$*　


```
CDSTIME$ (トラック番号)
```

機　能　CD中の各曲の開始時間を返します。

説　明　CD中の各曲の開始時間を、“分：秒：フレーム”の文字列で返します。

<トラック番号>……知りたい情報の入っているトラックを整数で指定します。

●指定できるトラック番号の範囲は、1からCDINFで得た曲数までです。その他の範囲を指定するとエラーになります。

関連命令　CDSTAT：CDドライブの状態を返します。
CDINF　：CDの内容を調べます。

CHAIN

チェイン *chain* 　　　　プログラムの分岐

```
CHAIN 〔MERGE〕 ファイルディスクトリプタ
  〔,〔行番号式〕〔, ALL〕〔, DELETE 範囲〕〕
```

機　能　指定されたファイルのプログラムを呼び出します。このとき現在メモリにあるプログラムから、変数を引き渡すことができます。

説　明　<MERGE>……現在メモリにあるプログラムに、ファイル上のプログラムを、行番号順に混ぜ合わせるときに指定します。
アスキー形式およびバイナリ形式のどちらのプログラムでも呼び出せます。ただし、バイナリ形式の場合には、ファイル上のプログラムの先頭行番号は、メモリ上のプログラムの最大行番号より大きくなくてはなりません。メモリ上のプログラムが使用しているファイルは、オープンされた状態のままで保たれます。

<ファイルディスクリプタ>……呼び出すプログラムが保存されているファイルを指定します。(☞「●ファイルディスクリプタの形式」p.31参照)

<行番号式>……プログラムの実行開始行番号を指定します。式を指定したときは、式の結果が実行開始行番号になります。
省略すると、プログラムの最初の行から実行します。<行番号式>の指定は、RENUM命令を実行しても変わりません。

<ALL>　……現在メモリにあるプログラムの全ての配列または変数を引き渡します。
ただし、FIELD命令で指定された文字変数は引き渡されません。
省略すると、COMMON命令で指定された変数または配列だけを引き渡します。

<DELETE> … <MERGE>の指定がされている時のみに有効で、混ぜ合わせる前に現在メモリにあるプログラムの行を削除します。

- 呼び出し元および呼び出し先のプログラムは、ソースプログラムでなければなりません。
- ＜MERGE＞を省略すると、それまでのプログラムで行っていた変数の型の指定は無効になります。したがってこのような場合、DEFINT、DEFLNG、DEFSNG、DEFDBLおよびDEFSTRの各命令は、CHAIN命令によって連結されたプログラムの中で、必要に応じてそれぞれ再実行をしてください。また、それまでのプログラムで使用していたファイルはクローズされます。
- CHAIN命令の実行により、割り込み処理ルーチンおよびDEF FN命令の定義は無効になります。連結されたプログラムの中で再実行をしてください。
- ＜DELETE　範囲＞により行われるプログラムの行削除の形式は以下のとおりです。

```
DELETE  ( { 行番号1   } ) ( { - } ( { 行番号2   } ) )
        ( { ラベル名1 } ) ( { , } ( { ラベル名2 } ) )
```

・DELETE　行番号1－行番号2

＜行番号1＞または＜ラベル名1＞と、＜行番号2＞または＜ラベル名2＞を指定したときは、その間に含まれる行を削除します。

・DELETE　行番号1－

＜行番号1＞または＜ラベル名1＞と、コンマ（,）またはハイフン（－）のみを指定したときは、＜行番号1＞または＜ラベル名1＞以上の行番号を持つ全ての行を削除します。

・DELETE　行番号1

＜行番号1＞または＜ラベル名1＞のみを指定したときは、その行だけを削除します。

・DELETE　－行番号2

コンマ（,）またはハイフン（－）と、＜行番号2＞または＜ラベル名2＞のみを指定したときは、先頭の行から＜行番号2＞または＜ラベル名2＞までの行を削除します。

・DELETE　－

コンマ（,）またはハイフン（－）のみを指定したときは、全ての行を削除します。

CHR$関数

キャラクタ・ダラー関数　*character$*

文字列処理

```
CHR$ (式〔, 式〕…)
```

機　能　キャラクタコードを文字に変換します。

説　明　<式>で指定したキャラクタコードを、対応するANK文字に変換します。

<式>……キャラクタコードを指定します。

例　題　&H41から&H5Aまでの26個の数値をキャラクタコードとする文字を表示します。

```
1000 FOR I=&H41 TO &H5A
1100   PRINT CHR$(I);
1200 NEXT I
1300 END

結果：ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

関連命令　ASC関数 ：CHR$関数とは逆に、文字をキャラクタコードに変換します。

KNJ$関数: JISコードを対応する日本語文字に変換します。

お詫びと訂正

このたびは、Ｆ－ＢＡＳＩＣ３８６　Ｖ２．１、Ｆ－ＢＡＳＩＣ３８６コンパイラ　Ｖ２．１をお買上げいただき誠にありがとうございます。

Ｆ－ＢＡＳＩＣ３８６　Ｖ２．１　ガイド、Ｆ－ＢＡＳＩＣ３８６　Ｖ２．１　リファレンスにおきまして、不備がございました。深くお詫び申し上げますとともに、次のとおり訂正します。

☆Ｆ－ＢＡＳＩＣ３８６　Ｖ２．１　ガイドの２４２ページ下から９行目と１０行目の以下の記述を削除します。

削除内容
ＰＡＴＨ命令でパスが設定してあるときは、次のように入力し、パスを解除しておく必要があります。　PATH:⏎

☆Ｆ－ＢＡＳＩＣ３８６　Ｖ２．１　リファレンスの１１７ページの記述に以下の記述を追加します。

追加内容

コンパイラ
- ●行番号１、行番号２の指定は、ＲＥＮＵＭ命令を実行すると付け直されます。
- ●コンパイラでは、<ＤＥＬＥＴＥ範囲>の指定はできません。
- ●コンパイラでは、<ＭＥＲＧＥ>指定はできません。但し、メモリ上のプログラムが使用しているファイルは、オープンされたままの状態が保たれます。
- ●コンパイラでは、<ファイルディスクリプタ>に変数を指定することはできません。
- ●コンパイラでは、指定されたファイルのプログラム全体を呼び出します。<行番号式>で実行開始行番号を指定することはできません。
- ●コンパイラで、１つの実行形式プログラム中に複数のＣＨＡＩＮ命令があるとき、あるＣＨＡＩＮで<ＡＬＬ>を指定したら、そのプログラム中の他のＣＨＡＩＮでも<ＡＬＬ>を指定する必要があります。

関連命令
- ＲＵＮ　　　：メモリにあるプログラムを実行します。
- ＣＯＭＭＯＮ：ＣＨＡＩＮ命令により連結されるプログラムに引き渡す変数を指定します。

LE81-0115-1(81SP-0442-1)

CINT

コンバート・インテジャ *convert to integer*　計算

C

```
CINT（式）
```

機　能　ロング型整数値、単精度実数値、倍精度実数値を整数値に変換します。

説　明　<式>で指定する数値の小数部分を四捨五入して整数値に変換します。

<式>……ロング型整数値、単精度実数値または倍精度実数値を指定します。
指定できる範囲は、−32768 ～32767 です。

例　題　入力された単精度実数値、倍精度実数値を、整数値に変換し表示します。

```
1000 INPUT A
1100 A=CINT(A)
1200 PRINT A
1300 PRINT
1400 INPUT B#
1500 B=CINT(B#)
1600 PRINT B
1700 END

結果：? 16.369
      16
      ? 985.23645
      985
```

関連命令　CDBL：整数値、ロング型整数値、単精度実数値を倍精度実数値に変換します。
CLNG：整数値、単精度実数値、倍精度実数値をロング型整数値に変換します。
CSNG：整数値、ロング型整数値、倍精度実数値を単精度実数値に変換します。

CIRCLE

サークル *circle*

画面表示

1

```
         ┌ (wx,wy)     ┐
CIRCLE  ─┤             ├─, 半径 [,〔色〕 [,〔比率〕 [,〔開始位置〕
         └ STEP (x,y)  ┘

                                                  ┌ 中塗り色       ┐
 [,〔終了位置〕 [, [ ┌F┐ ] [,〔論理操作〕 [,   ─┤ タイルストリング ├─ ]]]]]]]
                     └N┘                          └ ラインスタイル   ┘
```

機　能　中心点と半径を指定して、円、円弧、またはだ円を描きます。

説　明　＜wx, wy＞　……円の中心点をワールド座標値で指定します。

＜STEP(x, y)＞…円の中心点を最終参照座標（LP）からの距離（x, y）で指定します。

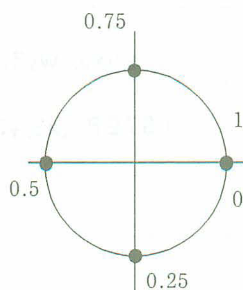
＜半径＞　……X軸方向の半径を、ワールド座標上の長さで指定します。

＜色＞　……円周の色を指定します。
省略すると，直前のCOLOR命令で指定した＜前景色＞で描かれます。
（☞「●色の指定」p.54参照）

＜比率＞……X軸方向とY軸方向の半径の比率を、ドット数の比で指定します。
Y軸方向の半径の値は＜半径＞×＜比率＞になります。
省略すると、１を指定したとみなされ、真円が描かれます。

＜開始位置＞……円を描き始める位置を0～1の数値で指定します。

＜終了位置＞……円を描き終える位置を0～1の数値で指定します。



0～1の値は、左の図の位置を示します。
<開始位置>＝<終了位置>のとき、完全な円が描かれます。それ以外は、<開始位置>から<終了位置>までの円弧を描きます。
<開始位置>または<終了位置>を省略すると、0を指定したとみなされます。

<F><N> ……円や円弧の内部を<中塗り色>または<タイルストリング>で塗りつぶすか否かを、指定します。

F：塗りつぶします。

N：塗りつぶしません。

省略すると、Nを指定したとみなされます。

Fを指定したときに、<中塗り色>および<タイルストリング>の指定を省略すると、<色>で塗りつぶされます。

Nを指定したとき、または指定を省略したときに<ラインスタイル>を指定すると、指定した線種で円周または円弧を描くことができます。

<論理操作> ……色についての操作（PSET、PRESET、AND、OR、XOR、OPAQUEまたはPASTEL）を指定します。
（☞「●グラフィック命令の論理操作」p.60参照）
省略すると、PSETを指定したとみなされます。

<中塗り色>……円や円弧の内部を塗りつぶす色を指定します。

<タイルストリング>……円や円弧の内部を塗りつぶすタイルパターンを指定します。（☞「●パターンの指定」p.61参照）

<ラインスタイル>……円周の線の種類を指定します。（指定についての詳細は、LINE命令を参照してください。）

●CIRCLE命令実行後のLP（最終参照座標）は、<wx,xy>になります。

2

```
CIRCLE {(wx1, wy1)   } - {(wx2, wy2)   } - {(wx3, wy3)   }
       {STEP (x1, y1)}   {STEP (x2, y2)}   {STEP (x3, y3)}
〔,〔色〕〔,〔論理操作〕〔,ラインスタイル〕〕〕
```

機　能　指定した3点を通る円または円弧を描きます。

説　明　<(wxn, wyn)>……円弧の始点、通過点、終点をワールド座標値で指定します。始点=終点のときは、全円になります。

<STEP(xn, yn)>……円弧の始点、通過点、終点を最終参照座標（LP）からの距離（x, y）で指定します。LPは、点を指定するごとにその座標に移っていきます。すなわち、<STEP(x2, y2)>は<(wx1, wy1)>からの距離、<STEP(x3, y3)>は<(wx2, wy2)>からの距離を示します。

<色>……円周の色を指定します。省略すると、直前のCOLOR命令で指定した<前景色>で描かれます。（☞「●色の指定」p.54参照）

<論理操作>……色についての操作（PSET、PRESET、AND、OR、XOR、OPAQUEまたはPASTEL）を指定します。（☞「●グラフィック命令の論理操作」p.60参照）省略すると、PSETを指定したとみなされます。

<ラインスタイル>……円周の線の種類を指定します。（指定についての詳細は、LINE命令を参照してください。）

●CIRCLE命令実行後のLP（最終参照座標）は、<wx3,wy3>になります。

関連命令 LINE　：線また長方形を描きます。
PAINT：指定した色で囲まれた範囲を別の指定した色で塗りつぶします。

CLEAR

クリア　*clear*　　　　一般命令

```
CLEAR〔, ,〔スタック領域の大きさ〕〔,〔配列変数領域の大きさ〕
 〔,〔プロシジャ領域の大きさ〕〔, DLL領域の大きさ〕〕〕〕
```

機　能　全ての数値変数を0に、全ての文字変数を空文字列に初期設定し、BASICが使用するメモリの領域の大きさを設定します。

説　明　現在メモリ内にあるプログラムを除いた全てのメモリを開放します。全ての数値変数を0に、全ての文字変数を空文字列に設定し、配列変数を消去します。
また、＜スタック領域の大きさ＞、＜配列変数領域の大きさ＞および機械語プログラムを読み込む＜プロシジャ領域の大きさ＞、＜DLL領域の大きさ＞を設定することができます。

＜スタック領域の大きさ＞……BASICがFOR～NEXTまたはGOSUB命令などで使用するスタックの大きさを256バイト以上の偶数バイト数で指定します。
省略すると、大きさは変わりません。

＜配列変数領域の大きさ＞……配列変数に割り当てられる領域の大きさを偶数バイト数で指定します。
省略すると、大きさは変わりません。

＜プロシジャ領域の大きさ＞…LOADM命令によってプロシジャを読み込むための領域の大きさを偶数バイト数で指定します。
省略すると、大きさは変わりません。

＜DLL領域の大きさ＞ ………DLLを使用した拡張命令のためにDLL領域の大きさをバイト数で指定します。
省略すると、大きさは変わりません。
DLLを使用している命令とその必要サイズは次のとおりです。

MOVIE：500Kバイト以上(アニメーションファイルの場合)
　　　400Kバイト以上(動画ファイルの場合)
DEF FONT　：300Kバイト以上
JPEGファイル：300Kバイト以上

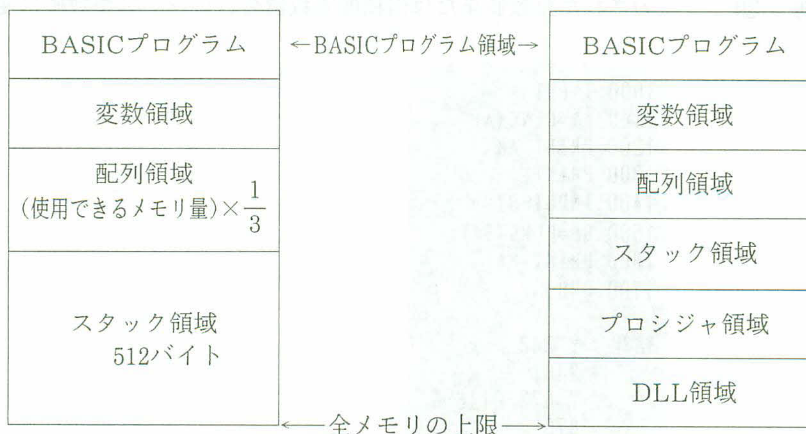
- DEF命令（DEF FN、DEFINT、DEFLNG、DEFSNG、DEFDBLまたはDEFSTR）により定義された全ての情報は、無効になります。
- 各大きさの指定で奇数バイトを指定すると、偶数に繰り上げられます。
- 各領域の大きさは、BASIC起動時には次のように初期設定されています。

  スタック領域の大きさ………512バイト

  配列変数領域の大きさ………“使用できるメモリ量”の1／3の大きさ

  プロシジャ領域の大きさ……0バイト

  DLL領域の大きさ　…………0バイト
- “使用できるメモリ量”はBASIC起動時に画面に表示されます。
- V1.1のBASICでは、DLL領域の大きさは指定できません。



コンパイラ
- コンパイラでは、CLEAR命令で指定できるのはプロシジャ領域とDLL領域の大きさだけです。これ以外の指定は無視され、他の領域および変数には影響しません。

例　題　スタック領域を1Kバイトに、配列領域を4096バイトに指定します。

```
CLEAR,, 1024, 4096
```

関連命令　FRE関数：メモリ未使用領域の大きさを返します。

CLNG

コンバート・ロング　*convert to long*　　計算

```
CLNG (式)
```

機　能　整数値、単精度実数値または倍精度実数値をロング型整数値に変換します。

説　明　<式>で指定する数値の小数部分を四捨五入してロング型整数値に変換します。

<式>……整数値、単精度実数値または倍精度実数値を指定します。
指定できる範囲は、−2147483648～+2147483647です。

例　題　入力された整数値または倍精度実数値を、ロング型整数値に変換し表示します。

```
1000 INPUT A
1100 A&=CLNG(A)
1200 PRINT A&
1300 PRINT
1400 INPUT B#
1500 B&=CLNG(B#)
1600 PRINT B&
1700 END

結果：? 243
      243
      ? 873.0146
      873
```

関連命令　CDBL: 整数値、ロング型整数値または単精度実数値を倍精度実数値に変換します。
CINT：ロング型整数値、単精度実数値または倍精度実数値を整数値に変換します。
CSNG: 整数値、ロング型整数値または倍精度実数値を単精度実数値に変換します。

CLOSE

クローズ　*close*

データ入力・データファイル

C

CLOSE　〔〔#〕ファイル番号〔,　〔#〕ファイル番号〕…〕

機　能　ファイルのクローズ処理を行います。

説　明　OPEN命令で割り当てられたバッファの内容を出力し、<ファイル番号>の解除を行います。

<#>　　　　　……省略しても意味は変わりません。

<ファイル番号>……OPEN命令時に割り当てた番号を指定します。
省略した場合は、オープンされている全てのファイルがクローズされます。

- CLOSE命令を実行すると、そのファイルの入出力バッファを他のファイルで使用できるようになります。
- END、NEW、RUNまたはSYSTEM命令を実行するか、エディタ中にファイルが読み込まれると、オープンされているファイルが全てクローズされます。



BREAKキーなどでプログラムの実行を中断した場合には、必ずオープンされているファイルをクローズしてください。ファイルをクローズせずにフロッピィディスクを抜き出して別のフロッピィディスクに入れ換えると、新たに入れたフロッピィディスクの内容が読めなくなる恐れがあります。

例　題　オープンしたファイルをクローズします。

```
1000 OPEN "I",#1,"KYBD:"
1100 OPEN "O",#2,"LPT0:"
1200 INPUT#1,D$
1300 IF D$="END" THEN GOTO 1600
1400 PRINT#2,D$
1500 GOTO 1200
1600 CLOSE#1,#2                 ' オープンした#1,#2 をクローズする
1700 END
                                    プリンタへの出力
結果：? FUJITSU        →             FUJITSU
      ? 386AV                        386AV
      ? SERIES                       SERIES
      ? END
```

関連命令　OPEN: ファイルのオープン処理を行います。

END ：プログラムの実行を終了し、オープンされている全てのファイルをクローズします。

CLS

シー・エル・エス *clear screen* 画面表示


```
CLS 〔消去範囲コード〕
```

機　能　画面をクリアします。

説　明　<消去範囲コード>で、消去する画面を指定します。

<消去範囲コード>……消去する画面を0～5の数値で指定します。

0：グラフィック画面上のビューポートとテキスト画面の両方を消去します。
テキスト画面のカーソルは、スクロールウィンドウのホームポジションに移動します。スクロールウィンドウがない場合は，非スクロール1画面のホームポジションに移動します。
（☞「スクロールウィンドウ」P.139）

1：テキスト画面中のスクロールウィンドウを消去します。

2：テキスト画面中の非スクロール1画面を消去します。

3：テキスト画面中の非スクロール2画面を消去します。

4：テキスト画面の全てを消去します。

5：グラフィック画面のビューポート内を消去します。
テキスト画面は消去しません。

省略すると、0を指定したとみなされます。



256色モードでは、テキスト画面とグラフィック画面が独立でないため、<消去範囲コード>の指定にかかわらず、どちらの画面も消去されます。

例　題　画面の０行から７行までを非スクロール１画面、８行から14行までをスクロールウィンドウ、15行以下を非スクロール２画面になる画面を設定し、文字列を表示します。
その後、スクロールウィンドウ、非スクロール１画面、非スクロール２画面の順にそれぞれの画面を消去していきます。

```
1000  CONSOLE  8,7
1100  LOCATE 0,0
1200   FOR J=1 TO 700:PRINT "非スクロール1";:NEXT J
1300  LOCATE 0,8
1400   FOR I=1 TO 900:PRINT " スクロール";:NEXT I
1500  LOCATE 0,15
1600   FOR H=1 TO 985:PRINT "非スクロール2";:NEXT H
1700  LOCATE 0,0
1800   FOR A=1 TO 2000:NEXT A:CLS 1   ' スクロール画面を消去する.
1900   FOR B=1 TO 2000:NEXT B:CLS 2   ' 非スクロール1画面を消去する.
2000   FOR C=1 TO 2000:NEXT C:CLS 3   ' 非スクロール2画面を消去する.
2100  CONSOLE 0,23,0
2200  END
```

関連命令　CONSOLE：スクロールウィンドウを設定します。

COLOR〔省略形COL.〕

カラー　*color*　　画面表示


```
COLOR 〔文字色〕〔,〔背景色〕〔,〔前景色〕〔,〔アトリビュート〕〕〕〕
```

機　能　テキスト画面、グラフィック画面の表示色および背景色を指定します。

説　明　<文字色>……テキスト画面に表示する文字の色を0〜15の数値で指定します。

0〜7を指定すると、以下の色で文字が表示されます。

8〜15を指定すると、表示される文字は反転表示になります。現在の背景色が文字の色となり、指定した色がその外側になります。

0：（8）	黒	4：(12)	緑
1：（9）	青	5：(13)	シアン
2：(10)	赤	6：(14)	黄
3：(11)	マゼンダ	7：(15)	白

省略すると、現在の<文字色>を指定したとみなされます。

<背景色>……グラフィック画面の背景色を、色の指定方法に従って指定します。
（☞「●色の指定」p.54参照）

<背景色>の指定は、COLOR命令の後にCLS命令を実行すると有効になります。

ただし、<文字色>で反転表示を指定した場合は、CLS命令を実行すると、背景色が<文字色>で指定した色になります。

省略すると、現在の<背景色>を指定したとみなされます。

<前景色>……グラフィック画面に表示する色を、色の指定方法に従って指定します。（☞「●色の指定」p.54参照）

省略すると、現在の<前景色>を指定したとみなされます。

＜アトリビュート＞……テキスト画面に表示される文字の表示機能を0, 1, 4または5のいずれかの数値で指定します。

0：ノーマル　（通常の表示）
1：リバース　（反転表示）
4：インテンシティ　（高輝度表示）
5：インテンシティリバース（高輝度反転表示）

省略すると現在設定されている＜アトリビュート＞を指定したとみなされます。

●BASICを起動したときは、COLOR 7,0,7,0に初期設定されています。



16色モードおよび32768色モードでは、パレットを変化させても文字の色は変化しませんが、256色モードではパレット変更の影響を受け、色が変わります。

例　題　文字色0〜7で、文字列を表示します。

```
1000 CLS
1100 FOR C=0 TO 7
1200   COLOR C
1300   PRINT
1400   PRINT"***** 文字色=";C;"*****"
1500 NEXT C
1600 COLOR 7
1700 END
```

関連命令　PALETTE：パレットの色を設定します。

COMMON

コモン　*common*　　　プログラムの分岐

C

```
COMMON　変数名〔, 変数名〕…
```

機　能　CHAIN命令により連結されるプログラムに引き渡す変数を指定します。

説　明　COMMON命令は、連結するプログラムに引き渡す変数を定義する非実行文です。ALL指定のないCHAIN命令と対で使用します。

<変数名>……CHAIN命令で連結されるプログラムに引き渡す変数を指定します。

- 1つまたは複数のCOMMON命令で、同じ変数を重複して指定することはできません。
- 配列変数を指定する場合には，変数名の後に“（）”を記述します。
指定した配列名が未定義の場合は、エラーになります。
- エラーが発生したときは、変数と配列はCLEAR命令が実行されたときと同じ状態に初期設定されます。
- COMMON命令は、プログラム中のどこに記述しても構いませんが、通常は、プログラムの先頭に記述します。
- CHAIN命令により連結するプログラムに、全ての変数を引き渡すときは、COMMON命令を使用するより、CHAIN命令でALL指定をする方が便利です。

コンパイラ
- コンパイラでは、CHAIN命令で連結されるプログラムの、呼び出し元と呼び出し先の両方のCOMMON命令の内容が、同じでなければなりません。

例　題　メインプログラムからサブプログラムへ変数を引き渡します。
サブプログラムで、渡された内容を画面に表示します。

```
メインプログラム
1000 A$="富士通株式会社"
1100 B#=1234567890#
1200 COMMON A$,B#              ' 引き渡す変数（A$, B#）を指定する
1300 CHAIN "COMMONSB.BAS"     ' 変数（A$, B#）を引き渡す
1400 END

サブプログラム (COMMONSB.BAS)
2000 PRINT A$
2100 PRINT B#
2200 END

結果：富士通株式会社
　　　1234567890
```

関連命令　CHAIN：指定したファイルからプログラムを呼び出します。

COM(n) ON/OFF/STOP

コム・オン/オフ/ストップ　*communication(n) on/off/stop*


```
                ┌ON ┐
COM(ポート番号) ┤OFF├
                └STOP┘
```

機　能　通信回線からの割り込みを許可、禁止または停止します。

説　明　<ポート番号>……通信ポート番号を0～4の数値で指定します。

0：本体内のRS−232Cポート
1：増設RS−232Cインタフェースの通信ポート1
2：増設RS−232Cインタフェースの通信ポート2
3：増設RS−232Cインタフェースの通信ポート3
4：増設RS−232Cインタフェースの通信ポート4

- COM (n) ON
  指定した通信ポートからの入力割り込みを許可します。割り込みが発生するとON COM (n) GOSUB命令で指定したルーチンが呼び出されます。
- COM (n) OFF
  指定した通信ポートからの入力割り込みを禁止します。
- COM (n) STOP
  指定した通信ポートからの入力割り込みを停止します。
  後にCOM (n) ONを実行すると、保留中の割り込みが実行されます。
- COM (n) ON／OFF／STOP命令を実行する前に，ON COM (n) GOSUB命令で、割り込み処理ルーチンを定義しておく必要があります。

例　題　ON　COM (n) GOSUB命令の例題を参照してください。

関連命令　ON　COM (n) GOSUB：通信回線からの割り込み処理ルーチンを定義します。

CONNECT

コネクト *connect* 　　　　画面表示

```
CONNECT [{(wx1, wy1)
         {STEP (x1, y1)}]  -{(wx2, wy2)
                             {STEP (x2, y2)}

[ -{(wx3, wy3)
   {STEP(x3, y3)} ] … [,〔色〕[,〔論理操作〕〔,{N 〔,ラインスタイル〕
                                          {F ,{中塗り色
                                              {タイルストリング}]]]]]
```

機　能　複数の点を直線で結びます。

説　明　指定したｎ個の座標間を、指定した順に直線で結びます。

＜wxn, wyn＞……直線の通過点を，ワールド座標値で指定します。

＜STEP(xn, yn)＞……直線の通過点を、最終参照座標（LP）からの距離（x, y）で指定します。
LPは、点を指定するごとにその座標に移り、＜STEP（x2, y2）＞は＜wx1, wy1＞からの距離、＜STEP（x3, y3）＞は＜wx2, wy2＞からの距離を示します。
省略すると、LPを指定したとみなされます。

＜色＞……結ぶ直線の色を指定します。
（☞「●色の指定」p.54参照）
省略すると、直前のCOLOR命令で指定した＜前景色＞で表示されます。

＜論理操作＞……色についての操作（PSET、PRESET、AND、OR、XOR、OPAQUEまたはPASTEL）を指定します。
（☞「●グラフィック命令の論理操作」p.60参照）
省略すると、PSETを指定したとみなされます。

＜N, ラインスタイル＞……線の種類を指定します。(指定のしかたは、LINE命令を参照してください。)

＜ラインスタイル＞を省略すると、実線で描かれます。

＜F, タイルストリング＞ ＜F, 中塗り色＞……結んだ直線の内側を、＜中塗り色＞または＜タイルストリング＞で塗りつぶします。

＜中塗り色＞は、色の指定方法に従って記述します。

(☞「●色の指定」p.54参照)

＜タイルストリング＞の指定のしかたは、「●パターンの指定」(☞ p.61) を参照してください。

＜中塗り色＞および＜タイルストリング＞を省略すると、＜色＞で塗りつぶされます。

- ●＜N＞または＜F＞の指定を省略すると、実線が描かれます。
- ●CONNECT命令実行後のLP(最終参照座標)は、最後に指定した値＜wxn, wyn＞になります。

例　題　いくつかの座標を指定し、直方体を描きます。

```
1000 CLS
1100 X=500:Y=250:D=60
1200 AX=X+0:AY=Y+0:BX=X-80:BY=Y+20:CX=X+10:CY=Y+40:DX=X+80:DY=Y+20
1300 CONNECT(AX,AY)-(BX,BY)-(CX,CY)-(DX,DY)-(AX,AY),5,PSET
1400 CONNECT(AX,AY+D)-(BX,BY+D)-(CX,CY+D)-(DX,DY+D)-(AX,AY+D),5,PSET
1500 CONNECT(AX,AY)-(AX,AY+D),2,PSET
1600 CONNECT(BX,BY)-(BX,BY+D),3,PSET
1700 CONNECT(CX,CY)-(CX,CY+D),4,PSET
1800 CONNECT(DX,DY)-(DX,DY+D),6,PSET
1900 END
```

関連命令　LINE：線または長方形を描きます。

CONSOLE〔省略形CONS.〕

コンソール *console* 　　画面表示

```
CONSOLE 〔スクロール開始行〕〔,〔スクロール行数〕〔,〔PFキー表示スイッチ〕〕〕
```

機　能　テキスト画面のスクロールウィンドウを設定します。

説　明　＜スクロール開始行＞と＜スクロール行数＞を指定して、スクロールウィンドウを設定します。

＜スクロール開始行＞……スクロールを開始する行を、0～24の数値で指定します。
省略すると、現在設定されているスクロール開始行を指定したとみなされます。

＜スクロール行数＞　……スクロールウィンドウの行数を0～25の数値で指定します。
省略すると、現在設定されているスクロール行数を指定したとみなされます。

＜PFキー表示スイッチ＞……システム行の有無と、その行にPFキーを表示するか否かを0、1または2の数値で指定します。

0：システム行なし。PFキーを表示しません。
この状態では、INPUT文などの実行時に単語登録と再変換ができません。

1：システム行を画面の最終行に確保し、その行の左70桁に、PFキーの内容を表示します。

2：システム行を画面の最終行に確保しますが、PFキーを表示しません。

省略すると、現在の＜PFキー表示スイッチ＞を指定したとみなされます。

●BASIC起動時は、CONSOLE 0,25,0 に初期設定されています。

● <スクロール開始行>に0、または24を指定し、<スクロール行数>に0を指定したときは、全画面が非スクロール画面になります。
● <スクロール開始行>に1〜23を、<スクロール行数>に0を指定したときは、画面が2つの非スクロール画面に分割されます。
● CONSOLE命令を実行すると、テキスト画面の全てが消去され、カーソルが画面のホームポジションに移動します。画面が2つの非スクロール画面に分割されているときは、非スクロール2画面のホームポジションに移動します。

用語

・ホームポジションとは、先頭行左端位置をいいます。
・非スクロール画面とは、スクロールウィンドウでない部分のテキスト画面をいいます。
スクロールウィンドウの上下に非スクロール画面があるとき、上の画面を非スクロール1画面、下の画面を非スクロール2画面と呼びます。
・スクロールウィンドウの最終文字位置に文字を出力すると、スクロールウィンドウの第2行から最終行までが1行ずつ上方向にスクロールし、カーソルは空白になった最終行の左端に移動します。非スクロール画面の最終文字位置に文字を出力すると、その画面がクリアされ、カーソルは空白になった画面の先頭行の左端に移動します。

例　題　画面の上から第0行〜第9行を非スクロール1画面、第10行〜第14行をスクロールウィンドウ、第15行〜第24行を非スクロール2画面に設定します。

```
1000 CONSOLE 10,5
```

(行数)

行	画面
0〜9	非スクロール1画面
10〜14	スクロールウィンドウ
15〜24	非スクロール2画面

関連命令　WIDTH 80：スクロールウィンドウを解除し、テキスト画面をクリアします。

CONT〔省略形C.〕

コント　*continue*　　　　コマンド

```
CONT
```

機　能　中断しているプログラムの実行を再開します。

説　明　BREAKキー押下後、またはSTOP、END命令実行後、中断しているプログラムの実行を再開します。

- プログラムの実行停止後、一行実行でCONT命令を入力すると、停止した次の文から実行が再開されます。ただしINPUT命令で実行が停止したときは、その行の初めから実行が再開されます。
- 実行を中断して一行実行により中間結果を調べたり、変数の値を変更したりすることができます。その後CONTまたはGOTO命令により、プログラムの実行を再開することが可能です。
  ただし、プログラム中断中にプログラムを変更したときは、CONT命令により実行を再開することはできません。
- プリンタに出力中にBREAKキーが押されたときは、CONT命令により実行を再開することはできません。
- GOSUBで呼び出したサブルーチンをエラートラップ処理により中断した場合、CONT命令を使用することはできません。
- BASICエディタの「実行」の「Continue」を指定した場合と同じ結果になります。

コイパイラ
- コンパイラでは、CONT命令は使用できません。CONT命令を含むプログラムをコンパイルするとエラーになります。

関連命令　STOP：プログラムの実行を中断します。

COS関数

コサイン関数　*cosine*　　計算


```
COS（式）
```

機　能　三角関数コサイン（余弦）の値を返します。

説　明　<式>……角度（単位：ラジアン）を指定します。

ラジアン＝π×角度（°）／180（°）

●式の型が倍精度の場合は倍精度の、単精度の場合は単精度の値を返します。

例　題　入力した倍精度型の数値（°）の、三角関数コサインの値を求めて表示します。

```
1000 INPUT A
1100 B=COS(3.14159/180*A)              ' 単精度の値をB に与える
1200 PRINT B
1300 INPUT A#
1400 B#=COS(3.14159265359#/180*A#)     ' 倍精度の値をB#に与える
1500 PRINT B#
1600 END

結果：? 52
      .615662
      ? 42.23654798965
      .740375966621467
```

関連命令　SIN関数　：三角関数サイン（正弦）の値を返します。

TAN関数：三角関数タンジェント（正接）の値を返します。

CSNG

コンバート・シングル *convert to single*

計算

CSNG(式)

機　能　整数値、ロング型整数値または倍精度実数値を単精度実数値に変換します。

説　明　<式>で指定した数値を有効数値6桁の単精度実数値に変換します。

<式>……整数値、ロング型整数値または倍精度実数値を指定します。
指定できる範囲は、−3.40282E+38〜+3.40282E+38です。

例　題　入力された整数値、倍精度実数値を、単精度実数値に変換し表示します。

```
1000 INPUT A%
1100 A=CSNG(A%)
1200 PRINT A
1300 PRINT
1400 INPUT B#
1500 B=CSNG(B#)
1600 PRINT B
1700 END

結果：? 45.36
      45
      ? 256.32145
      256.321
```

関連命令　CDBL：整数値、ロング型整数値または単精度実数値を倍精度実数値に変換します。

CINT　：ロング型整数値、単精度実数値または倍精度実数値を整数値に変換します。

CLNG：整数値、単精度実数値または倍精度実数値をロング型整数値に変換します。

CSRLIN関数

カーソル・ライン関数　*cursor line*　　画面表示

C

```
CSRLIN
```

機　能　テキスト画面上のカーソルの垂直位置を返します。

説　明　カーソル位置の、キャラクタ座標のＹ座標値を返します。

例　題　カーソルの垂直位置と水平位置を画面に表示します。

```
1000  LOCATE 10,3
1100  A=CSRLIN                    ' カーソルの垂直位置をAに返す.
1200  B=POS(0)
1300  PRINT A,B
1400  END

結果：          3          10
```

関連命令　POS関数：テキスト画面上のカーソルの水平位置、またはプリンタバッファ内の水平文字位置を返します。

CVI/CVL/CVS/CVD関数

文字列処理・データファイル

シー・ブイ・アイ　*convert to integer*
シー・ブイ・エル　*convert to long*
シー・ブイ・エス　*convert to single*
シー・ブイ・ディ　*convert to double*

```
CVI (2バイトの文字列)
CVL(4バイトの文字列)
CVS (4バイトの文字列)
CVD(8バイトの文字列)
```

機　能　文字で表現された数値を、値が示す数値データに変換します。

説　明　ランダムファイルから読み込んだデータは、全て文字型になっています。
CVI、CVL、CVSおよびCVD関数は、文字型のデータを数値データに変換します。

CVI：2バイトの文字列を整数に変換します。
CVL：4バイトの文字列をロング型整数に変換します。
CVS：4バイトの文字列を単精度実数に変換します。
CVD：8バイトの文字列を倍精度実数に変換します。

● 単精度形式、倍精度形式の数値は、IEEE形式です。

例　題　キーボードから入力した数値を文字型に変換してランダムファイルに書き込みます。そのデータを読み出し、数値に変換して表示します。

```
1000 OPEN "R",#1,"SAMPLE"
1100 FIELD #1,8 AS D$
1200 INPUT "D#=";D#
1300 LSET D$=MKD$(D#)
1400 PUT #1,1
1500 GET #1,1
1600 D#=CVD(D$)
1700 PRINT D#
1800 CLOSE #1
1900 END

結果：D#=? 952.32145687
      952.32145687
```

```
1000 OPEN "R",#1,"SAMPLE"
1100 FIELD #1,4 AS C$
1200 INPUT "C!=";C!
1300 LSET C$=MKS$(C!)
1400 PUT #1,1
1500 GET #1,1
1600 C!=CVS(C$)
1700 PRINT C!
1800 CLOSE #1
1900 END

結果：C!=? 65.3214
      65.3214
```

関連命令

MKI$／MKL$／MKS$／MKD$関数：数値を文字列に変換します。

CVSMBF／CVDMBF関数：マイクロソフト内部形式の文字列を数値データに変換します。

STR$関数：数値を数字の文字列に変換します。

VAL関数：数字の文字列を数値に変換します。

CVSMBF/CVDMBF関数

シー・ブイ・エス・エム・ビー・エフ関数　*convert to single Microsoft binary format*　文字列処理・データファイル
シー・ブイ・デイ・エム・ビー・エフ関数　*convert to double Microsoft binary format*

```
CVSMBF (4バイトの文字列)
CVDMBF(8バイトの文字列)
```

機　能　マイクロソフト内部形式の文字列で表現された値を、数値データに変換します。

説　明　F-BASIC86HG V1.2 以前のF-BASICで書き込まれたランダムファイルでは、データは全て旧形式（マイクロソフト内部形式）の文字型になっています。CVSMBFおよびCVDMBFは、その文字データを数値データに変換します。

CVSMBF：4バイトの文字列を単精度の実数に変換します。
CVDMBF：8バイトの文字列を倍精度の実数に変換します。

● 単精度形式、倍精度形式の数値は、IEEE形式です。

関連命令　MKSMBF$関数：倍精度型実数を文字列に変換します。
MKDMBF$関数：単精度型実数を文字列に変換します。

DATA

デ一タ　*data*　　　　データ入力

D

```
DATA　定数〔, 定数〕…
```

機　能　READ命令で読み込む数値、および文字定数を定義します。

説　明　<定数>……数値定数、または文字定数を指定します。

１つのDATA命令には、１行の範囲で複数個の<定数>をコンマで区切って記述することができます。

数値定数では、以下のものが指定できます。

整数、ロング型整数、固定小数点、浮動小数点、16進数、8進数

以下の文字定数を指定するときは、文字定数全体をダブルクォーテーション（"）で囲みます。

・先頭または末尾に空白のある文字列。

・コンマ、コロン、シングルクォーテーションを含む文字列。

- DATA命令は非実行文です。プログラムのどこにでもいくつでも記述することができます。DATA命令で定義した定数は、行番号の若いDATA命令の順に、一連の連続した定数として扱われます。
- DATA命令で定義した定数は、READ命令により順に読み込まれます。対応する変数と型が一致していなければなりません。
- RESTORE命令でDATA命令の記述された行を指定することにより、定義した定数を何度でも読み込むことができます。

例　題　DATA命令で定義した定数を、READ命令で読み込みます。D1には数値変数が、D2$には文字変数が読み込まれます。D1に0が読み込まれたらプログラムを終了します。

```
1000 READ D1,D2$                  '1400 行のDATAを変数D1, D2$ に読み込む
1100 IF D1=0 THEN END
1200 PRINT D1,D2$                 ' 変数D1, D2$ を画面に出力する
1300 GOTO 1000
1400 DATA 1,a,2,bb,3,ccc,0,dddd ' データを定義

結果：1          a
      2          bb
      3          ccc
```

関連命令　READ　：DATA命令により定義された定数を変数に読み込みます。
RESTORE：READ命令の読み込み対象になるDATA命令の位置を指示します。

DATE関数

デート関数　*date*　　時計

D

```
DATE
```

機　能　年初（1月1日）から現在までの日数を返します。

説　明　1月1日からBASICシステムタイマが示す日付までの日数を返します。

例　題　プログラム実行日（BASICシステムタイマの日付）と、1月1日からプログラム実行日までの日数を表示します。

```
1000 PRINT "今日は，";DATE$;"です．"      '今日の日付
1100 PRINT "1月1日から，";DATE;"日です．"  '  1月1日からの日数
1200 END
結果：今日は，89/03/18です．
      1月1日から，77 日です．
```

関連命令　DATE$　：BASICシステムタイマが示す日付を返すか、または日付を変更します。

TIME関数：BASICシステムタイマが示す時刻を秒で返します。

TIME$　：BASICシステムタイマが示す時刻を返すか、または変更します。

DATE$

デート・ダラー　*date$*　　時計

1　DATE$

機　能　BASICシステムタイマが示す日付を返します。

説　明　日付を、yy／mm／ddの形式で返します。

yy　：西暦年の下２桁
mm：月
dd　：日

●yyが80～99のときは1980～1999年を示し、00～79のときは2000～2079年を示します。

例　題　BASICシステムタイマの日付を、画面に表示します。

```
1000 PRINT DATE$                'ＤＡＴＥ＄命令を実行する

結果：90/01/01
```

2　DATE$= 文字式

D

機　能　BASICシステムタイマの日付を変更します。

説　明　<文字式>……変更する日付を文字列 "yy/mm/dd" で指定します。

yy　：西暦年の下２桁を指定します。
mm：月を指定します。
dd　：日を指定します。

- 1980～1999年を示すときは、yyに80～99を指定し、2000～2079年を示すときは、00～79を指定します。
- 変更した日付は、BASICの実行中だけ有効で、いったんBASICを終了すると、変更前の値に戻ります。

例　題　BASICシステムタイマが示す日付を表示し、その値を変更した後、新しく設定した日付を表示します。

```
1000  PRINT "今日は，";DATE$; "です．"  '既に設定済の日付を表示
1100  INPUT "正しく表示されていますか (Y/N)";ANS$
1200  IF ANS$="N" OR ANS$="n" THEN GOSUB *変更
1300  END
1400   *変更
1500  INPUT "次の形式で正しい日付を入力してください． yy/mm/dd";D$
1600  DATE$=D$                          'D$の数字をDATE$ に設定する
1700  PRINT "日付を";DATE$;"に設定しました．"
1800  RETURN
結果：今日は，90/01/03です．
      正しく表示されていますか(Y/N) ? N
      次の形式で正しい日付を入力して下さい．  yy/mm/dd? 90/01/04
      日付を90/01/04に設定しました．
```

関連命令　DATE関数：１月１日から現在までの日数を返します。
TIME関数　：BASICシステムタイマが示す時刻を秒で返します。
TIME$　　：BASICシステムタイマが示す時刻を返すか、または変更します。

DEF FN

ディファイン・ファンクション　*define function*　　一般命令

```
DEF FN 名前〔（パラメタリスト）〕＝式
```

機　能　関数の名前を付け、定義します。

説　明　<名前>……関数の名前を数値変数名または文字変数名で指定します。
関数名は、「FN+名前」となります。FNと名前の間に空白を入れてはいけません。
<名前>は、半角文字なら40文字以内、全角文字なら20文字以内で指定します。

<パラメタリスト>……この関数で使用する全ての変数を指定します。
変数は、関数を呼び出すときに与えられる引数と1対1に対応します。変数名の型は、引数の型と一致しなければなりません。
変数は、コンマで区切って記述します。

<式>……関数計算をする数値式を指定します。
関数名の型と<式>の結果の型は、一致していなければなりません。

- <パラメタリスト>に記述された変数名と同名の変数が、プログラムの他の行で使用されていてもかまいません。異なった変数として扱われます。
- CHAIN命令を実行すると、DEF FN命令で定義された関数は無効となりますので、連結されたプログラムの中で定義しなおす必要があります。

インタプリタ
- インタプリタでは、プログラムを実行する順序が、DEF FN命令で関数を定義してから、その関数が呼ばれるようにしなければなりません。また、エディタから一行実行でDEF FN命令を使うことはできません。

コンパイラ
- コンパイラでは、実行順序に関係なく、ソースプログラムをコンパイルしたときの、DEF FN命令の出現順序で関数が宣言されます。ソースプログラム上の出現順序と、プログラムの実行順序が異なる場合には、注意が必要です。

（☞「●DEF命令の機能の違い」p.517参照）

例　題　関数名FNABCと言う関数をDEF FN命令で定義します。次に引数を、5, 10, 20としてFNABCを呼び出します。

```
1000 DEF FNABC(X,Y,Z)=(X+Y)*Z
1100 ABC=FNABC(5,10,20)
1200 PRINT ABC
1300 END

結果: 300
```

DEF FONT

ディファイン・フォント *define font*

画面表示

```
DEF FONT "フォント名"
```

機　能　SYMBOL命令で表示する文字のフォントを指定します。

説　明　この命令以後に書かれたSYMBOL命令では、このフォントで、文字が表示されます。

＜フォント名＞………フォント名を指定します。

- SYMBOL命令の＜角度コード＞を０以外に設定した場合、DEF FONTの指定にかかわらず、システム16ドットフォントで描画されます。
- ベクトルフォントを使用する場合、ベクトルフォントカードが必要です。
  ベクトルフォントカードがないと、システム16ドットフォントで表示されます。
- 指定したフォントがなかった場合や、フォントを表示するのに必要なメモリが足りなかった場合、間違ったフォント名を指定した場合は、システム16ドットフォントで表示されます。
- 起動時はシステム16ドットフォントになっています。
- V1.1のBASICでは、DEF FONT命令は使用できません。
- CLEAR命令でDLL領域を設定していない場合は、システム16ドットフォント以外を指定するとエラーになります。

DEFINT/DEFLNG/DEFSNG/DEFDBL/DEFSTR

ディファイン・イント/ロング/シングル/ダブル/ストリング *define integer/long/single/double/string* 一般命令


```
DEFINT
DEFLNG
DEFSNG      文字の範囲〔，文字の範囲〕…
DEFDBL
DEFSTR
```

機　能　変数の型を宣言します。

説　明　<文字の範囲>で指定した文字で始まる全ての変数を、以下の型に宣言します。

DEFINT ……変数の型を整数に宣言します。
DEFLNG……変数の型をロング型整数に宣言します。
DEFSNG……変数の型を単精度実数に宣言します。
DEFDBL……変数の型を倍精度実数に宣言します。
DEFSTR……変数の型を文字に宣言します。

<文字の範囲>……変数を次のように指定します。

1文字の日本語文字	：指定した文字で始まる全ての変数。
1文字の英字（ANK文字）	：指定した文字で始まる全ての変数。
1文字の英字−1文字の英字	：アルファベット順で、指定した英字の間にある文字で始まる全ての変数。

- 「1文字の日本語文字−1文字の日本語文字」の指定はできません。
- 型宣言文字を持つ変数は、その型宣言がDEF命令（DEFINT／DEFLNG／DEFSNG／DEFDBL／DEFSTR）による宣言より優先されます。
- DEF命令による型宣言がなく、型宣言文字も伴わない変数は、全て単精度実数型の変数として扱われます。

インタプリタ ●インタプリタでは、実行順序にしたがって、型宣言が有効になります。

コンパイラ ●コンパイラでは、実行順序に関係なく、ソースプログラムをコンパイルしたときの、DEF命令の出現順序で宣言が有効になります。ソースプログラム上の出現順序と、プログラムの実行順序が異なる場合には、注意が必要です。
（☞「●DEF命令の機能の違い」p.517参照）

例　題

DEFINT A,C,E	A,C,Eで始まるすべての変数を整数変数に宣言します.
DEFLNG A-C	AからCまでの文字で始まるすべての変数をロング型整数変数に宣言します.
DEFSNG B-X	BからXまでの文字で始まるすべての変数を単精度変数に宣言します.
DEFDBL D-G	DからGまでの文字で始まるすべての変数を倍精度変数に宣言します.
DEFSTR A,E-G	A,E,F,Gで始まるすべての変数を文字変数に宣言します.
DEFINT 年,月,日	年，月，日で始まるすべての日本語変数を整数変数に宣言します.

DEF KANJI

ディファイン・カンジ　*define kanji*　　画面表示

D

```
DEF　KANJI　外字コード，ドットパターン
```

機　能　外字コードを定義します。

説　明　16×16ドットからなる外字パターンを指定し、これに外字コードを割り当てます。

<外字コード>……外字を登録する位置を、４桁の16進数で指定します。指定できる数値は次の範囲です。

＆H7521～＆H757E
＆H7621～＆H767E

<ドットパターン>……定義する外字のドットパターンを指定します。ドットパターンは、16進数または文字列で指定します。

- 16進数指定の場合
  ＆Hで始まる16進数を64個ならべて指定します。
- 文字列指定の場合
  CHR＄関数を使い、32バイトの文字列で指定します。

- 最大188個の外字を登録できます。また、登録した外字は、PRINT、SYMBOL文で漢字コードとして使用することができます。
- ドットパターンが32バイトに満たない場合には、残りのバイトには&H00が入ります。また、32バイトを超えた分は無視されます。
- この命令を使用するときは、外字パターン領域を確保する必要があります。領域を確保しないと外字パターンが正しく表示されません。
- 外字パターンを再び使いたいときは、TownsMENUで外字ファイルを保存する必要があります。
- V1.1L20以前のBASICでは、DEF KANJI命令は使用できません。

DEF MOVIE

ディファイン・ムービー　*define movie*　　動画／アニメーション

1

```
DEF MOVIE 1,(X,Y)
```

機　能　動画を再生する左上座標を指定します。

説　明
- この命令は、ムービーファイルが動画ファイル(.MVE)、アニメーションファイル(.MMM) のどちらにも共通の命令です。
- V1.1のBASICでは、この命令は使用できません。

2 DEF MOVIE 2, SPEED

機　能　動画を再生する速度を制御します。

説　明　<SPEED>

省略	：再生速度を制御します。元の速度に対する伸長速度は1倍です。
0	：再生速度を制御しません。（この場合、フレーム抜けがありません。また、音がでません。）
0.1～4.0	：元の速度に対する再生速度を（割合）を設定します。 1未満でスロー再生になり、1以上で高速再生になります。 （高速またはスロー再生の場合は、音がでません。）

● この命令は、ムービーファイルが動画ファイル(.MVE)の場合にだけ使えます。アニメーションファイル(.MMM)に対して実行すると、エラーになります。

● V1.1のBASICでは、この命令は使用できません。

3 DEF MOVIE 3, VOLUME

機　能　動画再生時のPCMの音量を設定します。

説　明　＜VOLUME＞……PCMの音量を設定します。

- この命令は、動画ファイル(.MVE)に音が入っている場合にだけ使えます。
- この命令は、ムービーファイルが動画ファイルの場合にだけ使えます。
  アニメーションファイル(.MMM)に対して実行すると、エラーになります。

DEF PEN

ディファイン・ペン　*define pen*

画面表示

D

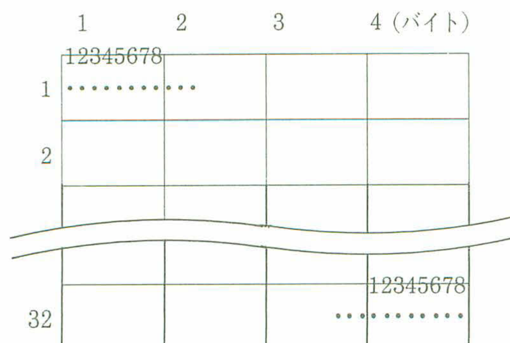
DEF　PEN　{ 0〔, ペンの太さ〕 / 1〔, 配列名〕 }

機　能　線や点を描画するのに使用するペンの太さ、ペンの形状を設定します。

説　明　<ペンの太さ>……描画する線の幅（単位：ドット）を0～32の範囲で指定します。0を指定すると描画されません。
省略すると、1を指定したとみなされます。

<配列名>　……ペンの形状（32×32ドットのドットパターン）を示す配列変数を指定します。<配列名>は、DIM文で定義された4バイト×32要素の配列の変数名です。
省略すると、直前に設定したペンの形状を指定したとみなされます。

●ペン形状は次のようになります。左上の8ビットから右下の8ビットまでの128バイトの領域で、順にビットパターンを指定します。これは、ロング型整数変数の要素32からなる一次元配列と対応付けられます。ただし、32ドット×32ドットの四角形に内接する円の外側のドットは、1が指定されても表示されません。



●BASIC起動時および以下の命令を実行したときは、DEF PEN　0, 1になります。

```
SIMPOSE ON/OFF
SINPUT
SPRITE ON/OFF
VIEW
SCREEN@
```

例題 1　線の太さを10ドットに指定して、(100, 100) から (200, 200) まで線を引きます。

```
1000 DEF PEN  0,10
1100 LINE(100,100)-(200,200),PSET
```

例題 2　32バイトの配列にドットパターンを設定します。(100, 100) を中心とした直径32ドットのまだら模様を描きます。

```
1000 DIM PAT&(31)
1100 A&= &H0F0F0F0F
1200 FOR I=0 TO 31 STEP 2
1300   PAT&(I)=A&
1400   PAT&(I+1)=NOT A&
1500 NEXT
1600 DEF PEN 1,PAT&
1700 PSET (100,100),5
```

DEF SPRITE

ディファイン　スプライト　*define sprite*　　画面表示

スプライトの各種情報を定義します。

D

1
```
DEF  SPRITE  0, パターン番号, 配列名, モード
```

機　能　スプライトパターンを定義します.

説　明　16×16ドットからなるスプライトパターンにパターン番号を割り当てます。
また、そのパターンが16色モードか、32768色モードかを指定します。

<パターン番号>……スプライトパターンに割り当てる番号を、数字で指定します。
指定できる数字は、モードにより以下のように制限されます。
16色モード　　:0～895の範囲で、任意の数字を指定します。
32768色モード:0～892の範囲で、任意の４の倍数を指定します。



16色モードと32768色モードのスプライトパターンの<パターン番号>が重複しないように注意してください。
32768色モードの<パターン番号>は４の倍数で指定しますが、そのパターンを制御するために<パターン番号>とそれに続く３個の番号を使用します。したがって、それらの番号は16色モードの<パターン番号>として使用することができません。
たとえば<パターン番号>に８を指定した場合、９、10および11も使われます。

<配列名>……16ドット×16ドットのスプライトパターンの配列を指定します。文字型の配列は指定できません。

＜モード＞……パターンが16色モードか、32768色モードかを指定します。

0：16色モード

スプライトパターンごとに32,768色中の任意の16色を選んで使用できます。

1：32768色モード

スプライトパターンでごとに32,768色中の任意の256色を選んで使用できます。

●実際の作成方法やパターンの構成等は、「●スプライトキャラクタの作成」（p.65）を参照してください。

●DEF SPRITE命令で定義したスプライトパターンやキャラクタの定義情報は、その情報をクリアするかBASICを終了するまで有効です。

2
```
DEF SPRITE 1, キャラクタ番号, (sx, sy), 先頭パターン番号〔,
〔横スプライト数〕〔,〔縦スプライト数〕〔,〔オフセット参照〕〔,
色テーブル番号〕〕〕〕
```

機　能　スプライトキャラクタを定義します。

説　明　スプライトキャラクタは、1個以上のスプライトパターンからなり、SPRITE命令で操作する単位です。

<先頭パターン番号>から始まる複数個のスプライトパターンに、<キャラクタ番号>を割り当てます。

<キャラクタ番号>……スプライトキャラクタに割り当てる番号を、0～1023の任意の数字で指定します。

<sx, sy>……スプライトキャラクタの初期表示位置を、スプライトキャラクタの左上隅のドットを表示する位置のスプライト座標値で指定します。

<先頭パターン番号>……スプライトパターンの<パターン番号>を指定します。この<パターン番号>から始まる<横スプライト数>×<縦スプライト数>個のパターンで、スプライトキャラクタを定義します。

<横スプライト数>……スプライトキャラクタを構成するスプライトパターンの横方向のパターン数を1～32の数値で指定します。
省略すると、1を指定したとみなされます。

<縦スプライト数>……スプライトキャラクタを構成するスプライトパターンの縦方向のパターン数を、1～32の数値で指定します。
省略すると、1を指定したとみなされます。

＜オフセット参照＞……スプライトキャラクタをオフセット参照の対象とするかどうかを、0または1で指定します。
(☞用語「オフセット参照」p167参照)
0：参照の対象としない。
1：参照の対象とする。
省略すると、0を指定したとみなされます。

＜色テーブル番号＞……16色モードのスプライトキャラクタの場合、使用する色テーブル番号を、0〜255の数値で指定します。
(☞「●色テーブル」p.68参照)
省略すると、0を指定したとみなされます。



＜色テーブル番号＞の指定は,16色モードのスプライトキャラクタでのみ有効です。

- ●キャラクタ番号の小さいスプライトキャラクタほど，画面上の表示優先順位が高くなります。つまり、画面上でスプライトキャラクタが重なったときには、キャラクタ番号の値の大きいキャラクタが下に隠れます。
- ●また、キャラクタ番号は小さい数値を使用した方が、実際のスプライト操作時に動きが高速になります。たとえば、キャラクタ番号として100を使用するよりも、0を使用する方が動きが速くなります。

注意

複数のスプライトパターンからなるスプライトキャラクタを定義する場合、与えるキャラクタ番号は1つですが、そのキャラクタ番号に続く連番が、指定したパターンの数だけキャラクタ番号として使用されます。したがって、キャラクタ番号を指定するときは、この点に注意して番号が重複しないようにします。

スプライトキャラクタ

0	4
8	12

0〜12は、32768色モードのスプライトパターン

たとえば、左図のような2×2の4パターンから成るスプライトキャラクタを定義する場合、<キャラクタ番号>を0にすると、連続する1、2および3も使用されます。したがって、別のスプライトキャラクタの<キャラクタ番号>には、4以上の番号を指定しなければなりません。

用語

「オフセット参照」

SPRITE命令の形式7（移動）でスプライトキャラクタを移動させるときにキャラクタ番号を省略すると、オフセット参照の対象となっているスプライトキャラクタが移動します。

つまり、複数のキャラクタをオフセット参照の対象に設定しておくと、SPRITE命令でそれらを同時に移動させることができます。

3 DEF SPRITE 2, 色テーブル番号, 配列名

機　能　色テーブルを定義します。

説　明　指定した配列名を持つ色テーブルに、＜色テーブル番号＞を割り当てます。16色モードのスプライトパターンにのみ有効な機能です。

（☞「●色テーブル」p.68参照）

＜色テーブル番号＞……色テーブルに割り当てる固有の番号を、0～255の数値で指定します。

＜配列名＞　……色テーブルを定義した配列を指定します。

例　題　星を点滅させます。32個の色テーブルの要素1だけを32通りの輝度で表示します。

```
1000 ' 16 COLOR SPRITE
1100 SCREEN@ 0
1200 DIM A%(150),C%(16)          ' パターンA%と色テーブルC%の配列宣言
1300 SYMBOL(0,0),"★",1,1,%1
1400 GET@A(0,0)-(15,15),A%
1500 DEF SPRITE 0,0,A%,0
1600 DEF SPRITE 1,0,(0,80),0,1,1,0,0
1700 DEF SPRITE 1,1,(240,150),0,1,1,0,31
1800 FOR I=0 TO 31:C%(1)=I*2^5+I:DEF SPRITE 2,I,C%:NEXT
1900 SCREEN@ 1
2000 SPRITE ON
2100 SPRITE 0,0,1
2200 SPRITE 0,1,1
2300 FOR X=0 TO 240
2400   SPRITE 2,0,X/8
2500   SPRITE 2,1,31-(X/8)
2600   FOR I=0 TO 50:NEXT
2700   SPRITE 6,0,1,0
2800   SPRITE 6,1,-1,0
2900 NEXT
```

4 DEF SPRITE 3〔, 色テーブル番号〕

機 能 色テーブルとしてパレット情報を指定します。(SCREEN@ 0 のときのみ)

説 明 16色モードのグラフィック画面で設定されているパレット情報を、そのまま色テーブルとして定義し、<色テーブル番号>を割り当てます。

<色テーブル番号>……色テーブルに割り当てる番号を、0〜255の数値で指定します。
省略すると、0を指定したとみなされます。

●スプライトキャラクタの色テーブルに、形式4で定義した色テーブル番号を選ぶと、そのスプライトパターンの間接色番号の0〜15をパレット番号の0〜15に対応付けます。(☞「●色テーブル」p.68参照)

5

```
DEF SPRITE 99,0
```

機　能　スプライト定義情報をクリアします。

説　明
●不要となったスプライトパターンやキャラクタの定義情報が多く残っていると、スプライトの動作が遅くなります。そのような場合はいったんクリアしてから、新しい定義情報だけを定義して使用すると、高速に動作させることができます。
●この命令を実行すると、いままでにDEF SPRITE命令で定義された全ての情報がクリアされます。

関連命令

SPRITE ON/OFF	：テキスト画面とスプライト画面の切り換えを行います。
SPRITE SCREEN	：グラフィック画面のどの位置にテキスト画面を表示するかを指定します。
SPRITE	：スプライトキャラクタを操作します。
SPRITE関数	：指定したスプライトキャラクタの情報を返します。
SPRITE TIME	：スプライト表示のタイミングを取り，複数パターンからなるスプライトキャラクタの移動時にパターンが乱れないようにします。

DELETE

デリート　*delete*　　コマンド

DELETE 〔{行番号1 / ラベル名1}〕〔{－ / ,}〕〔{行番号2 / ラベル名2}〕

機　能　プログラムの行を削除します。

説　明　<行番号1> <行番号2>…………削除する範囲を行番号で指定します。

<ラベル名1> <ラベル名2>……削除する範囲を行のラベル名で指定します。

●DELETE　行番号1－行番号2
<行番号1>または<ラベル名1>と、<行番号2>または<ラベル名2>をハイフン（－）またはコンマ（,）でつないで記述した場合は、その2つの行番号またはラベルにはさまれた行を削除します。

●DELETE　行番号1－
<行番号1>または<ラベル名1>の後ろに、ハイフン（－）またはコンマ（,）のみを記述した場合は、<行番号1>または<ラベル名1>以降の全ての行を削除します。

●DELETE　行番号1
<行番号1>または<ラベル名1>のみを記述した場合は、指定した行のみ削除します。

●DELETE　－行番号2
ハイフン（－）またはコンマ（,）に続けて<行番号2>または<ラベル名2>を記述した場合は、プログラムの先頭の行から<行番号2>または<ラベル名2>までの行を削除します。

●DELETE　－
ハイフン（－）またはコンマ（,）のみを記述した場合は、プログラムの全ての行を削除します。

●DELETEだけを記述すると、エラーになります。

●プログラムの中でDELETE命令を実行した場合、実行後、プログラムを終了し、BASICエディタに戻ります。

コンパイラ ●コンパイラでは、DELETE命令は使用できません。DELETE命令を含むプログラムをコンパイルするとエラーになります。

例　題　プログラム内の指定した行を削除します。

```
DELETE 100-500    ' 行番号100 から 500までを削除
DELETE -200       ' 先頭から行番号 200までを削除
DELETE 300-       ' 行番号 300から終わりまでを削除
DELETE 100        ' 行番号 100を削除
DELETE -          ' プログラム全体を削除
```

DIM

ディメンション　*dimension*　　一般命令

DIM　変数名（添字の最大値　〔,添字の最大値〕…）
〔,変数名（添字の最大値〔,添字の最大値〕…)〕…

機　能　配列変数を宣言し、メモリ領域を割り当てます。

説　明　<変数名>　……配列の変数名を指定します。

<添字の最大値>……配列の大きさを指定します。
（　）の中の<添字の最大値>の個数が、何次元の配列かを指定します。

- 宣言を行った直後は、指定された配列の全ての要素は、数値変数では0に、文字変数では空文字列に設定されます。
- DIM命令で宣言をしないで配列変数を使ったときは、<添字の最大値>は10に設定されます。
- 同一変数名の配列は宣言できません。宣言を更新する場合は、ERASE命令で以前の宣言を消去してから、改めて宣言します。

例　題　添字の最大値が、それぞれ11の2次元配列の宣言を行い、1から11までの乗算の結果を出力します。

```
1000 DIM A(11,11)
1100 FOR I=1 TO 11
1200   FOR J=1 TO 11
1300     A(I,J)=I*J
1400   NEXT J
1500 NEXT I
1600 FOR I=1 TO 11
1700   FOR J=1 TO 11
1800     PRINT USING "####";A(I,J);
1900   NEXT J:PRINT
2000 NEXT I
2100 END
```

関連命令　ERASE：配列変数をメモリから消去します。

DSKF関数

ディスク・エフ　*diskfile*

DSKF（ドライブ番号）

機　能　ディスクの未使用領域の大きさを返します。

説　明　<ドライブ番号>……ドライブを指定します。
ドライブは、システムの構成により、0～16まで指定できます。

●返される未使用領域の大きさは、キロバイト（KB）で示します。

例　題　ドライブ番号0のフロッピィディスクの未使用領域の大きさを表示します。

```
1000 A=DSKF(0)
1100 PRINT A
1200 STOP

結果:
     560
```

ELSE

エルス　*else*　　　　　　　　　　　　　　　　　　プログラムの分岐

```
ELSE〔コメント〕
```

E

機　能　IF～THEN命令で指定した式が偽の場合に実行する命令を定義します。

説　明　●ELSE命令の後ろに、コロン（：）で区切って２つ以上の命令文を記述することはできません。

詳細はIF～THEN命令を参照してください。

関連命令

IF～THEN	：式の結果を判定し、IFブロック内の処理を選択します。（構造化IF）
ENDIF	：IFブロックの終わりを示します。
ELSE IF～THEN	：式の結果を判定し、次の処理を選択します。

ELSE IF～THEN

エルス・イフ・ゼン　*else if then*　　　　プログラムの分岐

```
ELSE IF  <式>  THEN〔コメント〕
```

機　能　式の結果を判定し、次の処理を選択します。

説　明　●ELSE IF～THEN命令の後ろに、コロン（：）で区切って2つ以上の命令文を記述することはできません。

詳細はIF～THEN命令を参照してください。

関連命令

IF～THEN　：式の結果を判定し、IFブロック内の処理を選択します。（構造化IF）

ENDIF　：IFブロックの終わりを示します。

ELSE　：IF～THEN命令で指定した式が偽の場合に実行する命令を定義します。

END

エンド　*end*


```
END
```


機　能　プログラムの実行を終了し、オープンされている全てのファイルをクローズします。

説　明　実行中のプログラムを終了させ、ファイルのクローズ処理を行います。
実行後は、BASICエディタに戻ります。

- END命令は、プログラム中のどこにでも記述できます。また、複数のEND命令の記述も可能です。
- END命令の記述のないプログラムも実行可能ですが、その場合には実行終了時にファイルのクローズ処理は行われません。別途CLOSE命令によるファイルのクローズ処理の記述が必要になります。

例　題　プログラムを終了させます。

```
1000 PRINT "富士通"
1100 PRINT "株式会社"
1200 END
1300 PRINT "実行しません"

結果：富士通
　　　株式会社
```

関連命令
STOP　：プログラムの実行を中断します。
RUN　 ：プログラムを実行します。
CLOSE：ファイルのクローズ処理を行います。

ENDIF

エンド・イフ　*end of if*　　　　プログラムの分岐

```
END IF 〔コメント〕
```

機　能　IFブロックの終わりを示します。

説　明　●END IF命令の後ろに、コロン（：）で区切って２つ以上の命令文を記述することはできません。

詳細はIF〜THEN命令を参照してください。

関連命令

IF〜THEN　：式の結果を判定し、IFブロック内の処理を選択します。（構造化IF）

ELSE　：IF〜THEN命令で指定した式が偽の場合に実行する命令を定義します。

ELSE IF〜THEN　：式の結果を判定し、次の処理を選択します。

EOF関数

エンド・オブ・ファイル関数　*end of file*

データ入力

E

EOF（ファイル番号）

機　能　ファイルの終わりを検出します。

説　明　<ファイル番号>で指定した入力ファイルのEOF（End Of File）を検出し、検出結果を数値で返します。

<ファイル番号>……次の入力ファイルのいずれかをファイル番号で指定します。
- ・シーケンシャルファイル
- ・RS−232C
- ・キーボード

- ●EOF関数を実行する前に、入力ファイルはOPEN命令でオープンしておく必要があります。
- ●検出結果によって、−1または0を返しますが、返す値の意味は入力ファイルの種類によって異なります。

入力ファイル	検出結果	
	真(−1)	偽（0）
シーケンシャルファイル	EOFを検出した	EOFを検出していない
RS−232C キーボード	バッファが空	バッファが空でない

例　題　　シーケンシャルファイル“SAMP”からデータを読み込み、画面に表示します。

```
100  OPEN "O", #1,"SAMP"
110  READ A
120  WHILE A>0
130     PRINT #1,A
140     READ A
150  WEND
160  CLOSE #1
170  DATA 1236,987654,954,-1
1000 OPEN "I", #1,"SAMP"
1100 INPUT #1,A
1200 PRINT A
1300 IF NOT EOF(1) THEN 1100
1400 CLOSE #1
1500 END

結果：1236
      987654
      954
```

関連命令　LOF：入力バッファ中の文字数またはランダムファイルの最大レコード番号を返します。

ERASE

イレーズ *erase*　　一般命令

ERASE　配列変数名〔, 配列変数名〕…

機　能　配列変数をメモリから消去します。

説　明　<配列変数名>……消去したい配列変数名を指定します。

- ERASE命令で配列変数を消去した後は、DIM命令で同一変数名の配列を宣言することができます。
- 配列変数を消去すると、その変数に割り当てられていたメモリ領域は開放されます。

例　題　行番号1000で宣言した配列変数のうち、A、B、Cは行番号1100のERASE命令で消去されているので、新たに配列宣言することができます。Dは消去されていないため、行番号1500でDを配列宣言すると、エラーになります

```
1000 DIM A(100),B(100),C(100),D(100)
1100 ERASE A,B,C
1200 DIM A(50)
1300 DIM B(50)
1400 DIM C(50)
1500 DIM D(50)
```

関連命令　DIM：配列変数を宣言し、メモリを割り当てます。

E

ERR/ERL

エラー・ナンバー/エラー・ライン　*error number/error line*　　エラー処理

```
{ERR}
{ERL}
```

機　能　発生したエラーのエラー番号および行番号を返します。

説　明　ERRは直前に発生したエラーのエラー番号を、ERLは直前にエラーの発生した行の行番号を返します。

- 一般にERRとERLは、ON ERROR GOTO命令で指定されたエラー処理ルーチン内でエラーを判定し、プログラムの流れを制御するために使用します。
- RESUME命令を実行すると、ERRは0になりますが、ERLは次にエラーが発生するまで毎回そのエラー発生行番号を返します。
- 一行実行で実行された命令文でエラーが発生したときは、ERLは65535を行番号として返します。
- ERLは、READ命令の実行で発生したエラーの行番号を次のように返します。インタプリタでは、DATA命令の行番号を返します。コンパイラでは、READ命令の行番号を返します。

例　題　入力した数値が３桁以上ならエラーとして、エラー処理ルーチンを実行します。エラー処理ルーチンでは、エラーが発生した行の行番号を画面に表示します。

```
1000 ON ERROR GOTO *ERREXIT
1100 PRINT "２桁までの数字を入力してください．"
1200 *INPUT
1300   INPUT A
1400   IF A>99 THEN ERROR 99
1500   PRINT "O K !"
1600   END
1700 *ERREXIT
1800   IF ERR=99 THEN PRINT USING"( 行番号####)";ERL
1900   RESUME *INPUT
2000   END

結果：２桁までの数字を入力してください．
      ? 100
      ( 行番号1400 )
      ? 10
      O K !
```

関連命令

ON ERROR GOTO：エラー処理ルーチンを定義し、エラートラップを許可します。

RESUME　：エラー処理ルーチンからメインプログラムへ戻り、プログラム実行を再開します。

ERROR　：エラー発生をシミュレートします。

ERROR

エラー　*error*　　　　エラー処理

```
ERROR　エラー番号
```

機　能　エラー発生をシミュレートします。

説　明　＜エラー番号＞……1～255までの整数を数値式で指定します。
BASICのエラー番号にない番号を指定すると、ユーザプログラム独自のエラー番号として使用することができます。

● ERROR命令を実行すると、ON ERROR GOTO命令が指定されている場合は指定された行番号に分岐します。また、指定のない場合はエラー番号に対応するエラーメッセージが出力され、プログラムの実行が停止します。

どちらの場合もERROR命令が実行されると、変数ERRおよびERLにエラー番号とエラーが発生した行の行番号が代入されます。

例　題

```
1100 ON ERROR GOTO 1800
1200 RANDOMIZE (TIME/4)
1300 A=INT(RND(1)*9)+1
1400 INPUT B
1500 IF A<B THEN ERROR 80 ELSE IF A>B THEN ERROR 81
1600 PRINT "そのとおり！！"
1700 END
1800 IF ERR=80 THEN PRINT "大きすぎ！" ELSE PRINT "小さすぎ！"
1900 RESUME 1400
2000 END
結果：? 4
　　　大きすぎ！
　　　? 2
　　　小さすぎ！
　　　? 3
　　　そのとおり！！
```

関連命令	ERR/ERL	：発生したエラーのエラー番号および行番号を返します。
	ON ERROR GOTO	：エラー処理ルーチンを定義し、エラートラップを許可します。
	RESUME	：プログラムの実行を再開します。

EXP関数

イクスポネンシャル関数　*exponential*　　計算

```
EXP (式)
```

機　能　eを底とする指数関数の値を返します。

説　明　<式>……以下に示す正の値を指定します。

単精度のとき:　88.7229未満

倍精度のとき: 709.7827未満

●式の型が倍精度の場合は倍精度の、単精度の場合は単精度の結果を返します。

例　題　入力した倍精度型の数値の、指数関数を求めて表示します。

```
1000 INPUT A
1100 B=EXP(A)
1200 PRINT B
1300 INPUT A#
1400 B#=EXP(A#)
1500 PRINT B#
1600 END

結果：? 36.2
      5.26575E+15
      ? 15.32697854
      4533380.83367582
```

関連命令　LOG関数：式の自然対数を返します。

FIELD

フィールド　*field*　　　　　　　　　　　　　データファイル


```
FIELD〔#〕ファイル番号，フィールド幅　AS　文字変数名
　　　〔，フィールド幅　AS　文字変数名〕…
```

機　能　ランダムファイルのバッファを分割し、変数の領域を割り当てます。

説　明　＜ファイル番号＞で指定したランダムファイルのバッファを、＜フィールド幅＞で指定したフィールドに分割します。

＜＃＞　　　　　……省略しても意味は変わりません。

＜ファイル番号＞……オープン時にファイルに割り当てたファイル番号を指定します。

＜フィールド幅＞……バッファ内の各フィールドの長さをバイト数で指定します。指定できる範囲は1～255です。

＜文字変数名＞　……文字変数を定義します。

- GET命令またはPUT命令によりランダムファイルへの入出力処理を行う場合は、FIELD命令によりランダムファイルのバッファをフィールドに分割していなければなりません。
- FIELD命令で定義した文字変数へのデータ代入は、必ずLSETまたはRSETにより行ってください。INPUT命令など、LSET/RSET以外の命令によって代入しても、値は保証されません。
- 1つのFIELD命令で指定した文字変数のフィールド幅の合計は、ファイルをオープンした際に指定したレコード長以下でなければなりません。
- 同じファイル番号に対して、複数個のFIELD命令を記述し、同じフィールドを違った変数名で参照することも可能です。
- FIELD命令で定義された文字変数の内容は、そのファイルがクローズされるまで保証されます。

例　題　オープンしたランダムファイルバッファに、二重にフィールドを定義します。行番号1100で20バイトをNAM$に、12バイトをTEL$に割り当てます。次に同じ32バイトの領域を行番号1200でNAMTEL$に再定義します。

```
1000 OPEN"R",#1,"DATA5DAT"          'ランダムファイルをオープン
1100 FIELD #1, 20 AS NAM$,12 AS TEL$ 'NAM$,TEL$ のフィールドを定義
1200 FIELD #1, 32 AS NAMTEL$        'NAMTEL$ のフィールドを定義
1300 LSET NAM$="電話の故障"
1400 LSET TEL$="113"
1500 PUT#1,1  'データを書き込む"
1600 GET#1,1  'データを読み出す"
1700 PRINT "       名　前：";NAM$
1800 PRINT "       電　話：";TEL$
1900 PRINT "名　前　電　話：";NAMTEL$
2000 CLOSE#1
2100 END

結果：        名　前：電話の故障
              電　話：113
        名　前　電　話：電話の故障          113
```

関連命令　LSET／RSET：文字データを、左詰めまたは右詰めでランダムファイルバッファのフィールドに代入します。

FILES

ファイルズ　*files*

データ入力・データファイル

```
FILES 〔ファイルディスクリプタ〕〔,L〕
```

F

機　能　指定したデバイスのディレクトリリストを出力します。

説　明　＜ファイルディスクリプタ＞で指定したデバイスのディレクトリリストを、画面またはプリンタに出力します。

＜ファイルディスクリプタ＞……対象にするデバイスを指定します。ファイル名にワイルドカードの指定もできます。
（ 「ワイルドカード」p.34参照）
ファイルディスクリプタのファイル名を省略すると、指定されたディレクトリ配下のディレクトリリストが出力されます。
ファイルディスクリプタを省略すると、カレントドライブのカレントディレクトリリストが出力されます。
ファイルディスクリプタの＜オプション＞は指定できません。

＜L＞……Lを指定すると、ディレクトリリストがプリンタに出力されます。
省略すると、画面に出力されます。

- ディレクトリリストには、ファイル名、ファイルの大きさ、作成年月日の情報があります。ファイルの大きさは、バイト単位で表されます。
- ファイル名一覧は、BASICエディタのファイルの「読み込み」などでファイルウィンドウに表示されますが、FILES命令を使うと、全体を一度に見ることができます。

FIX

フィックス *fix*

計算

```
FIX（式）
```

機　能　＜式＞の小数点以下を切り捨て、整数部分を返します。

説　明　＜式＞……数値を指定します。

●FIX（x）は、SGN（x）＊INT（ABS（x））と同じ結果になります。
●FIX（x）とINT（x）の値は, xが正のときは同じですが、負のときは異なります。xが負のとき、FIX（x）はxの値より大きい整数になります。INT（x）は、xの値より小さい整数になります。

例　題　入力した数値の整数部分を表示します。FIX（−52.3）とINT（−52.3）では値が異なります。

```
1000 INPUT A
1100 B=FIX(A)      ' 入力された式の整数部分を変数B に与える.
1200 C=INT(A)
1300 PRINT B
1400 PRINT C
1500 END

結果 : ? -52.3
       -52    ' FIX 命令使用
       -53    ' INT 命令使用
```

関連命令　INT：式の結果を超えない最大の整数値を返します。

FOR

フォー *for*


```
FOR  制御変数＝初期値  TO  終値〔STEP  増分値〕
```

機　能　NEXT命令とループを作り、制御変数の値を数えながら一連の命令を繰り返し実行します。

説　明　<制御変数>……ループのカウンタとして使用する数値変数を指定します。倍精度型変数、配列変数の要素は指定できません。

<初期値>　……ループ初期の制御変数の初期値を、数値式で指定します。

<終値>　　……ループを終了する値を、数値式で指定します。

<増分値>　……<初期値>に加えられる値を、数値式で指定します。省略すると、1を指定したとみなされます。

- ループの実行中に、<初期値>、<終値>および<増分値>の値を変えてはいけません。
- 実行は次のように行われます。

①<初期値>を<制御変数>の値として、FOR命令からそれに対応するNEXT命令までの一連の命令文を実行します。

②NEXT命令に達すると<制御変数>の値に<増分値>を加えた値を<制御変数>の新しい値として、<終値>と比較します。

③<制御変数>が<終値>より小さいとき（<増分値>が負なら、<制御変数>が<終値>より大きいとき）は、FOR命令の次の行に戻り、同じ処理を繰り返します。これをFOR～NEXTループといいます。

④<制御変数>が<終値>より小さくないとき（<増分値>が負なら<制御変数>が<終値>より大きくないとき）は、ループをぬけてNEXT命令の次の行を実行します。

- ＜初期値＞が＜終値＞より小さくないとき（＜増分値＞が負なら＜初期値＞が＜終値＞より大きくないとき）、FOR～NEXTのループは1回だけ実行されます。
- ＜増分値＞に0を指定すると、無限にループが繰り返されます。
- FOR～NEXTループの中に別のFOR～NEXTループを作成し、入れ子構造にすることができます。
  また、入れ子構造を持つ異なるループは、それぞれ異なる＜制御変数＞を持つ必要があります。

例　題　制御変数を更新してNEXTまでの命令を繰り返し実行します。
行番号1000のFOR命令と行番号1500のNEXT命令、行番号1100のFOR命令と、行番号1300のNEXT命令がそれぞれループを作ります。

```
1000 FOR J=1 TO 5      ' 1500行までの間を繰り返し実行する
1100   FOR I=1 TO J  ' 1300行までの間を繰り返し実行する
1200     PRINT "*";
1300   NEXT I
1400   PRINT
1500 NEXT J
1600 END

結果：*
      **
      ***
      ****
      *****
```

関連命令　NEXT　：FOR～NEXTループの終わりを示します。
WHILE：一連の命令を条件付きで繰り返し実行します。

FRE関数

フリー関数 *free*　　　　一般命令

```
FRE（機能）
```


機　能　メモリの未使用領域の大きさを返します。

説　明　BASICで使用可能なメモリのうち、未使用領域の大きさを返します。

＜機能＞……メモリの領域を１、３または４のいずれかの数値で指定します。

１：テキスト領域の未使用領域のバイト数を返します。
この領域は、プログラムテキスト、単純変数および文字列の格納領域として使われます。

３：配列変数領域の未使用領域のバイト数を返します。

４：未使用テキスト領域（1）と、未使用の配列変数領域（3）を加えたバイト数を返します。

●＜機能＞に1または4を指定したときは、“ガーベッジコレクション（garbage collection）”と呼ばれる文字領域の整理を行います。整理後、文字領域の中の不要な文字列は消去され、必要な文字列のみが残ります。

コンパイラ　●コンパイラでは、＜機能＞の指定に関わりなく、BASICで使用可能な未使用領域の大きさを返します。

例　題　テキスト領域および配列変数領域の未使用領域のバイト数およびその合計を表示します。

```
1000 PRINT FRE(1)
1100 PRINT FRE(3)
1200 PRINT FRE(4)
1300 END
```

関連命令　CLEAR：全ての変数や定義を消去します。

GET

ゲット　*get*

画面表示・データファイル

```
GET〔#〕ファイル番号〔,レコード番号〕
```

機　能　ランダムファイルから、データをバッファに読み込みます。

説　明　＜ファイル番号＞で指定したランダムファイルから、＜レコード番号＞で指定した1レコードを、バッファに読み込みます。

＜＃＞……省略しても意味は変わりません。

＜ファイル番号＞……オープン時にランダムファイルに割り付けたファイル番号を指定します。

＜レコード番号＞……ランダムファイルの何番目のレコードを読み込むかを指定します。
省略すると、直前にそのファイルにGETまたはPUTしたレコードの次のレコードが読み込まれます。OPEN命令直後は、ファイルの先頭レコードが読み込まれます。

- ＜ファイル番号＞で指定したファイルは、ランダム入出力モード（R）で既にオープンされている必要があります。
- 読み込まれるレコードの長さは、OPEN命令のファイルディスクリプタで指定したレコード長です。

例　題　ランダムファイル"DATA1DAT"からデータを読み込み、そのデータを表示します。

```
1000 OPEN "R",#1,"DATA1DAT"
1100 FIELD #1, 20 AS A$,12 AS B$
1200 LINE INPUT " 名前は         ";NAM$
1300 LINE INPUT " 電話番号は     ";TEL$
1400 LSET A$=NAM$
1500 LSET B$=TEL$
1600 PUT #1
1700 FOR I=1 TO LOF(1)
1800   GET #1,I
1900   PRINT A$, B$
2000 NEXT
2100 CLOSE #1
2200 END

結果：名前は        警察
      電話番号は    110
      警察  110
```

関連命令　PUT　：バッファの内容をランダムファイルに出力します。

FIELD：ランダムファイルのバッファを分割し、変数の領域を割り当てます。

GET@

ゲット・アットマーク　*get@*

画面表示

1

```
GET@ (sx1, sy2)-(sx2, sy2), 配列名〔, 色〔, 色〕…〕
```

機　能　グラフィック画面上のドットパターンを配列に読み込みます。

説　明　２つのスクリーン座標を結ぶ線を対角線とする四角形内のドットパターンを＜配列名＞で指定した配列に読み込みます。

＜sx1, sy1＞＜sx2, sy2＞……四角形の対角線上の２点をスクリーン座標で指定します。
スクリーン座標値はVRAMの範囲を超えて指定することができます。

＜配列名＞……DIM命令で定義されている配列の変数名を指定します。
文字型の配列は指定できません。

＜色＞……読み込むドットパターンの色を指定します。
＜色＞と一致しているドットに対応する配列のビットが１になります。
省略すると、背景色以外の全ての色を指定したとみなされます。

●配列には次の大きさが必要です。

INT ((INT ((x+7)／8)＊y+a−1)／a)

x：横方向ドット数

y：縦方向ドット数

a：配列の型により以下のように設定されます。

配列の型が整数の場合	：2
配列の型がロング型整数，または単精度実数の場合	：4
配列の型が倍精度型実数の場合	：8

例　題　ドットパターンを読み込んで表示します。GET@は、色などの属性を読み込むことはできません。PUT@で画面に表示する際に指定します。

```
1000  CLS
1100  SYMBOL (150,50), "富士通", 4, 2, 1
1200  DIM A%(INT((INT((192+7)/8)*40+2-1)/2))
1300  GET@ (150,50)-(341,89), A%          ' ドットパターンを読み込む
1400  PUT@ (150,100)-(341,139),A%,PSET,2
1500  END
```

2　GET@A (sx1, sy1)−(sx2, sy2), 配列名〔, オフセット〕

機　能　グラフィック画面上のドットパターンを、その色の情報と共に配列に読み込みます。

説　明　2つのスクリーン座標を結ぶ線を対角線とする四角形内のドットパターンを＜配列名＞で指定した配列に読み込みます。

＜sx1, sy1＞ ＜sx2, sy2＞……四角形の対角線上の2点をスクリーン座標で指定します。

＜配列名＞……DIM命令で定義されている配列の変数名を指定します。文字型の配列は指定できません。

＜オフセット＞……配列の途中から読み込むとき、読み込み開始位置を（配列の先頭から数えた要素の個数）−1で指定します。
省略すると、0を指定したとみなし、配列の先頭から読み込みます。
多次元配列の場合は、0〜（各次元の要素数の積）−1の値を指定できます。

●配列には次の大きさが必要です。

16色モード　　：INT（(INT（(x+7)／8）＊y＊4+a −1)／a）
32768色モード：INT（(2＊x＊y+a −1)／a）
256色モード　：INT（(x＊y+a −1)／a）

x ：横方向ドット数
y ：縦方向ドット数
a ：配列の型により以下のように設定されます。

配列の型が整数の場合	：2
配列の型がロング型整数, または単精度実数の場合	：4
配列の型が倍精度型実数の場合	：8

関連命令　PUT@：GET@命令で配列に読み込まれたドットパターンを画面に表示します。

GOSUB〔省略形GOS.〕

ゴー・サブ *go to subroutine* プログラムの分岐

GOSUB ｛行番号 / ラベル名｝

機　能　サブルーチンを呼び出し、サブルーチンの終了後はGOSUB命令の次の行またはRETURN命令で指定した行に復帰します。

説　明　＜行番号＞ ＜ラベル名＞……呼び出すサブルーチンの実行開始行を指定します。

例　題　行番号1000で入力された半径の数値を使って、サブルーチンで円周と面積を計算し、結果を画面に表示します。

```
1000 INPUT "半径は ";R
1100 GOSUB *計算              ' ラベル名 *計算のサブルーチンを呼出
1200 GOSUB 1800               ' 1800行のサブルーチンを呼出
1300 END
1400  * 計算
1500      円周=2*R*3.1416!
1600      面積=R*R*3.1416!
1700      RETURN
1800  ' 印刷
1900      PRINT " 円周";円周
2000      PRINT " 面積";面積
2100      RETURN

結果：半径は ? 100
      円周  628.32
      面積  31416
```

関連命令　RETURN：サブルーチンを終了し、呼び出したプログラムへ復帰します。

GOTO〔省略形GO.〕

ゴーツー　*goto*　　　　　　　　　　　　　　　　プログラムの分岐

GOTO ｛行番号｜ラベル名｝

機　能　指定した行番号、またはラベル名の行に分岐します。

説　明　<行番号>　<ラベル名>……分岐先の行を指定します。

●<行番号>または<ラベル名>の行に実行文があるときは、その文から実行されます。その文が非実行文のときは、それに続く実行文から実行します。

例　題　無条件に分岐します。このプログラムを終了させるときは、BREAKキーを押してください。

```
1000 INPUT "A=";A
1100 INPUT "B=";B
1200 PRINT "加算";A+B
1300 PRINT "減算";A-B
1400 PRINT "乗算";A*B
1500 PRINT "除算";A/B
1600 GOTO 1000

結果：A=?1000
      B=?5
      加算 1005
      減算 995
      乗算 5000
      除算 200
      A=?
```

関連命令　GOSUB：サブルーチンに分岐し、サブルーチン終了後は、GOSUB命令の直後の行に復帰します。

G

HARDC

ハード・コピー　*hard copy*　　　　コマンド

```
HARDC 〔機能〕〔,（cx1,cy1）－（cx2,cy2）〕
```

機　能　テキスト画面またはグラフィック画面のデータをプリンタに出力します。

説　明　テキスト画面またはグラフィック画面のデータを、出力範囲や階調の指定をしてプリンタに出力します。

＜機能＞……0～2の数値で指定します。
0：テキスト画面を出力します。
1：グラフィック画面を16階調で出力します。
2：グラフィック画面を階調なしで出力します。
省略したときは、0を指定したとみなされます。

＜cx1, cy1＞＜cx2, cy2＞……出力する範囲をキャラクタ座標で指定します。
2つの座標を対角線の両端とする四角形を、出力範囲とします。
省略すると、画面全体を出力します。

●SPRITE ON状態など、テキスト画面の存在しない画面モードでは、HARDC 0 の指定はできません。

注意

・HARDC命令を使う場合は、BASICを立ち上げる前に使用するデバイスドライバをメモリに常駐させておく必要があります。
デバイスドライバの組み込み方は、システムに添付されたマニュアルを参照して下さい。

・HARDC命令は、標準プリンタ以外のプリンタでは実行できないことがあります。

・カラープリンタを指定した場合は、<機能>で1または2を指定しても、階調ではなくカラーで出力されます。

・HARDC命令は、LAN（Local area net work）のプリンタサーバへは出力できません。TOWNS本体に直接接続されたプリンタへのみ出力可能です。

例　題

```
HARDC                        'テキスト画面全体を印刷します。
                              HARDC 0,(0,0)-(79,24)に同じ。
HARDC 1,(20,15)-(38,20)      'キャラクタ座標(20,15)(38,20)を対角とする
                              四角形に対応するグラフィック画面を印刷します。
```

関連命令

LPRINT：式の結果をプリンタに出力します。

HEX$

ヘキサ・ダラー　*hexa decimal$*

計算

HEX$（式）

機　能　整数を16進数の文字列に変換します。

説　明　<式>……−2147483648～+4294967295の範囲の整数値を指定します。

- 変換結果は"0"～"FFFFFFFF"になります。
- <式>が小数部分を含む場合は、内部でINT関数を実行し、整数化してから変換します。
- 負の値を指定すると、内部表現の補数が16進数に変換されるので、正の値になります。

例　題　入力された値を16進数に変換し表示します。

```
1000 INPUT A&
1100 B$=HEX$(A&)
1200 PRINT "&H";B$
1300 END

結果：? 56
      &H38
```

関連命令　OCT$：整数を８進数の文字列に変換します。

IF～THEN

イフ・ゼン　*if～then*　　　　プログラムの分岐

```
IF　式　THEN〔コメント〕
```

機　能　式の結果を判定し、IF～THEN命令と、それに対応するEND IF命令までの間（IFブロック）にある処理を選択します。（構造化IF）

説　明　<式>が真（値が０でない）なら、THENで定義された命令文を実行し、<式>が偽（値が０）なら、対応するELSE命令またはELSE IF～THEN命令を実行します。

<式>……論理式、関係式、算術式のいずれかを指定します。

- IF～THEN命令の後ろに、コロン（：）で区切って２つ以上の命令文を記述することはできません。
- 入れ子構造にすることができます。IFブロックの入れ子は127個までです。

```
if・・・then ─────────┐
    If・・・then ─────┐  │
           :          │  │
    else              │  │
           :          │  │
    end if ───────────┘  │
else if・・・then        │
           :             │
else                     │
           :             │
end if ──────────────────┘
           :
```

●ELSE命令およびELSE IF～THEN命令は省略することができます。
●V1.1のBASICでは、IF～THEN(構造化IF)命令は使用できません。

例　題　3つの入力データを比較して、最も大きな数値を画面に表示します。

```
1000  INPUT "もっとも大きいのはどれ";A,B,C
1100  IF A>B THEN
1200     IF A>C THEN
1300        PRINT A
1400     ELSE
1500        PRINT C
1600     ENDIF
1700  ELSE IF B>C THEN
1800        PRINT B
1900  ELSE
2000        PRINT C
2100  ENDIF
2200  END

結果：
      もっとも大きいのはどれ? 50,20,80
      80
```

関連命令

ENDIF	：IFブロックの終わりを示します。
ELSE	：IF～THENで指定した式の結果が偽の場合に実行する命令を定義します。
ELSE IF～THEN	：式の結果によって次の処理を選択します。

IF～THEN～ELSE

イフ・ゼン・エルス　*if～then～else*　　　　プログラムの分岐

```
              ┌ 文〔：文〕…  ┐ ┌      ┌ 文〔：文〕… ┐ ┐
IF 式 THEN  ┤ 行番号         ├ │ ELSE ┤ 行番号       ├ │
              └ ラベル名      ┘ └      └ ラベル名     ┘ ┘

              ┌ 行番号  ┐       ┌ 文〔：文〕〔：文〕… ┐
IF 式 GOTO  ┤         ├  ELSE ┤ 行番号               ├
              └ ラベル名 ┘       └ ラベル名             ┘
```

I

機　能　式の結果を判定し、その後の処理を選択します。

説　明　＜式＞　　……論理式、関係式または算術式を指定します。

＜文＞　　……実行する命令文を記述します。文は複数記述できます。

＜行番号＞　＜ラベル名＞……分岐先の行を指定します。

- ＜式＞の値が真（値が０でない）のときは、THENの次の＜文＞、あるいは＜行番号＞または＜ラベル名＞の行、もしくはGOTO命令が実行されます。
- ＜式＞の値が偽（値が0）のときは、THENまたはGOTO命令は無視されます。ELSEがある場合はELSE以下の＜文＞、あるいは＜行番号＞または＜ラベル名＞の行が実行され、ない場合は次の行の命令文が実行されます。
- IF～THEN～ELSE命令では、THENとELSEの間、またはELSEの後に更に何重にもIF～THEN～ELSE命令を記述することができます。これを入れ子構造といいます。このとき、IF命令中のTHENとELSEの個数が異なるときは、ELSEはその直前のTHENに対応します。

例　題　3つの入力データを比較して、最も大きな数値を画面に表示します。

```
1000 INPUT A,B,C
1100 IF A>B THEN IF A>C THEN *X ELSE *Z ELSE IF B>C THEN *Y ELSE *Z
1200 *X
1300   PRINT A:END
1400 *Y
1500   PRINT B:END
1600 *Z
1700   PRINT C:END

結果：？25, 1, 900
      900
```

INKEY$関数

インキー・ダラー関数　*input key$*

データ入力

```
INKEY$
```

機　能　キーボードが押されていればその文字を、押されていなければ空文字を返します。

説　明　キーボードバッファが空でないときはバッファの先頭から1バイトを取り出し、その文字を返します。キーボードバッファが空のときは、空文字列を返します。

- BREAKキー以外のコントロールキャラクタも読み込みます。
- 読み込んだ文字は表示しません。

例　題　キーボードから入力された文字を表示し、"*"が入力されたらプログラムを終了します。

```
1000 A$=INKEY$:IF A$="" THEN 1000
1100 PRINT A$;
1200 IF A$="*" THEN 1400          ' * が入力されたら，1400行に飛ぶ
1300 GOTO 1000
1400 END

結果：FKIS,VOZW;956DPC*
```

関連命令

INPUT	：キーボードからデータを読み込みます。
LINE　INPUT	：キーボードから1行のデータを読み込みます。
INPUT$関数	：指定された文字数の文字列を、ファイルから読み込みます。

I

INP関数

インプット関数　*input*

INP（I／Oアドレス〔, 型番号〕）

機　能　ハードウェアのI／Oレジスタからデータを読み込みます。

説　明　<I／Oアドレス>……読み込むデータの位置を、アドレスで指定します。指定できる値は&H0000～&HFFFFです。

<型番号>……読み込みの単位を1,2または4で指定します。それぞれの数値は、以下の意味を持っています。

1:BYTE(1 byte)単位で読み込みを行います。

2:WORD(2 byte)単位で読み込みを行います。

4:DWORD(4 byte)単位で読み込みを行います。

省略すると1を指定したとみなされます。

- <I／Oアドレス>は、範囲外を指定すると、エラーになります。
- 指定した型番号と異なる型のデータを読み込もうとすると、プログラムは正しく動作しません。
- V1.1 L20以前のBASICでは、INP関数は使用できません。

INPUT

インプット *input*　　　　データ入力

INPUT 〔"プロンプトメッセージ" {, / ;}〕 変数名〔, 変数名〕…

機　能　キーボードからデータを読み込み、変数に代入します。

説　明　キーボードからデータ（合計255バイト以下）を入力し、入力順に<変数名>に代入します。入力時には<プロンプトメッセージ>が表示され、入力待ち状態になります。

<プロンプトメッセージ>……入力を促すメッセージを指定します。省略すると疑問符のみ表示します。

<, >……メッセージの後にコンマを記述すると、疑問符（?）を表示しません。

< ; >……メッセージの後にセミコロンを記述すると、疑問符（?）を表示します。

<変数名>……数値変数または文字変数を指定します。

- データを複数入力する場合、データ間の区切りにコンマ（, ）またはコロン（: ）を入力します。
- 以下の文字を文字列の一部として入力する場合は、文字列全体（空白も含む）をダブルクォーテーション（"）で囲みます。
  - ・先頭または末尾に空白を含む文字列。
  - ・コンマ（, ）またはコロン（: ）を含む文字列。
- 入力されるデータの型と個数は、<変数名>の型と個数に一致していなければなりません。
- データは、合計255バイトまで入力できます。

I

●データ入力中に、BREAKキーまたはCTRL+Cキーが押されると、プログラムの実行を中断し、BASICエディタに戻ります。中断直後に実行継続（Continue）を指示すると、INPUT命令の行から実行が再開されます。

例　題　入力したデータの合計と平均を表示します。

```
1000 INPUT "データの個数は";N     ' データの個数を読み込む
1100 FOR J=1 TO N
1200   PRINT J;
1300   INPUT "データ";DA          ' N個目のデータを読み込む
1400   S=S+DA
1500 NEXT J
1600 H=S/N                        ' 平均を求める
1700 PRINT "平均=";H              ' 平均を出力する
1800 END

結果：データの個数は? 3
      1 データ? 15
      2 データ? 20
      3 データ? 30
      平均=  21.6667
```

関連命令　LINE INPUT：キーボードから1行のデータを読み込みます。
INKEY$関数：キーボードが押されていればその文字を、押されていなければ空文字を返します。
INPUT$関数：指定した文字数の文字列を、ファイルから読み込みます。

INPUT$関数

インプット・ダラー関数　*input$*

データ入力


```
INPUT$ (文字数〔,〔#〕ファイル番号〕)
```

機　能　キーボードまたはファイルから入力された文字を返します。

説　明　＜文字数＞……入力する文字列の長さをバイト数で指定します。
範囲は0～255です。

＜#＞……省略しても意味は変わりません。

＜ファイル番号＞……読み込むデータが格納されているファイルに、オープン時に割り当てたファイル番号を指定します。
省略すると、キーボードからの入力になります。

- ファイルは、INPUT$命令が実行される前に、入力モード (I) でオープンされている必要があります。
- キーボードバッファの内容が、＜文字数＞以上の場合は、先頭から＜文字数＞分の文字を取り出します。また、＜文字数＞以下の場合は、＜文字数＞以上の文字が入力されるまで入力待ち状態になります。
- キーボードからの入力のとき、カーソルも文字も画面には表示されません。
- BREAKキー以外のコントロールキャラクタも、そのまま読み取ります。
読み取った文字は表示されません。

例　題　指定した文字数の文字列をキーボードから入力して表示します。

```
1000 INPUT "何文字入力しますか";A
1100 B$=INPUT$(A)
1200 PRINT B$
1300 END

結果：何文字入力しますか? 5
      12345
```

関連命令

INPUT#　　　：ファイルからデータを読み込みます。

INPUT　　　 ：キーボードからデータを読み込みます。

LINE INPUT：キーボードから1行のデータを読み込みます。

INPUT#

インプット・シャープ　*input#*　　　　データ入力・データファイル

```
INPUT#ファイル番号, 変数名〔, 変数名〕…
```

機　能　ファイルからデータを読み込み，変数に代入します。

説　明　<ファイル番号>で指定した入力バッファから、データを取り出し、<変数名>で指定した変数に代入します。

<ファイル番号>……オープン時にファイルに割り当てたファイル番号を指定します。

<変数名>　……読み込んだデータを代入する文字変数または数値変数を指定します。

- ファイルは、INPUT#命令が実行される前に、入力モード（I）でオープンされている必要があります。
- 変数名は、読み込むデータの型と一致していなければなりません。
- <変数名>が文字変数のとき、読み込んだデータにコンマ（,）、コロン（:）、CR、LFのいずれかがあると、データの区切りとみなされます。データ前後の空白は無視されます。ただし、ダブルクォーテーション（"）で囲まれている空白はデータとして代入されます。
- <変数名>が数値変数のとき、読み込んだデータにコンマ（,）、コロン（:）、CR、LFまたは１個以上の空白のいずれかがあると、データの区切りとみなされます。データの区切り文字コードや空白は、変数に代入されません。最初のデータに先行する空白は無視されます。
- 通信回線からデータを受信する場合、割り込みが発生してデータが入力バッファに貯えられます。INPUT#、LINE INPUT#、INPUT$が実行されないため、まだ読み込まれていないデータでバッファが一杯になったときに次のデータを受信すると、受信続行が不可能になりエラーになります。



INPUT#命令実行中に入力バッファに読みとるべきデータがなくなると、データがバッファに入ってくるまで待ち状態になり、次の命令以降が実行されません。
したがってこのようなことを避けるためには、LOFまたはEOF関数によりバッファの状態を知ってから、INPUT#命令を実行するようにします。

例　題　PRINT#の例題で作成されたシーケンシャルファイル"DATA2DAT"からデータを読み込んで表示させます。

```
1000 OPEN "I", #1,"DATA2DAT"
1100 PRINT"＜品名＞                ＜金額＞"
1200 WHILE EOF(1)<>-1
1300   INPUT#1, HINMEI$, KINGAKU  'ファイルデータを変数に読み込む
1400   PRINT HINMEI$, KINGAKU
1500 WEND
1600 CLOSE#1
1700 END

結果：＜品名＞          ＜金額＞
        にんじん            45
        大根               100
        いも                30
```

関連命令　LINE INPUT#　：ファイルからデータを1行分読み込み、区切ることなく変数に代入します。
INPUT$　：指定した文字数の文字列を、ファイルから読み込みます。

INSTR関数

インストリング関数　*instring*

文字列処理

```
INSTR（〔検索開始位置，〕文字式１，文字式２）
```

機　能　文字列を検索して指定された文字を見つけ、その位置をバイト位置で返します。

説　明　<文字式1>で指定した文字列から<文字式2>で指定した文字列を探し、文字列の先頭文字が<文字式1>の先頭から何バイト目にあるかを返します。

<検索開始位置>……<文字式1>で指定した文字列の何番目の文字から検索を開始するか1～255までの数値で指定します。
省略すると、１を指定したとみなされます。

<文字式1>　……検索される文字列を指定します。
文字列の長さは0～255バイトです。

<文字式2>　……探し出す文字列を指定します。
文字列の長さは0～255バイトです。

- <文字式2>が見つからなかった場合は、0が返されます。
<文字式1>が空文字列の場合、および<検索開始位置>が<文字式1>より大きい場合も、0が返されます。
- <文字式2>が空文字列の場合は、<検索開始位置>が返されます。
- 文字を数える場合、日本語文字は１文字２バイト、ANK文字は１文字１バイトとします。

例　題　日本語文字とANK文字を文字列の1バイト目から検索して見つけた位置を表示します。

```
1000 A$="F-BASIC"
1100 B$="C"
1200 C=INSTR(A$,B$)    'A$から"C"の位置を検索する
1300 PRINT C
1400 END

結果：7
```

関連命令　KINSTR関数：文字列を検索し、その位置を文字数で返します。

SEARCH　：配列変数の要素の中から指定された値を探し、配列の先頭からの順番を返します。

INT関数

インテジャ関数　*integer*

計算

INT(式)

機　能　式の結果を超えない最大の整数を返します。

説　明　<式>……数値を指定します。

●FIX（x）とINT（x）の値は、xが正のときは同じですが、負のときは異なります。xが負のときFIX（x）はxの値より大きい整数値になりますが、INT（x）はxの値より小さい整数値になります。

例　題　入力した数値の整数部分を表示します。FIX（−52.3）とINT（−52.3）では値が異なります。

```
1000 INPUT A
1100 B=INT(A)      ' 入力された式を超えない最大の整数を変数B に与える.
1200 C=FIX(A)
1300 PRINT B
1400 PRINT C
1500 END

結果：? -95.6
      -96
      -95
```

関連命令　FIX：式の結果の小数点以下を取り去り、整数部分を返します。

I

INTERVAL

インターバル　*interval*

時計

INTERVAL　割り込み間隔

機　能　インターバルタイマ割り込みの間隔を指定します。

説　明　＜割り込み間隔＞……1～65535の範囲の数値で秒数を指定します。指定した値が整数値でないときは、小数点以下を切り捨てて整数に変換します。

- INTERVAL命令を実行すると、指定した秒間隔で割り込みが発生し、割り込み許可状態であれば、そのつどON INTERVAL GOSUB命令で定義された割り込み処理ルーチンへ制御が移ります。
- 割り込みはBASICの文の切れ目で発生するので、実際の割り込み間隔は、指定した＜割り込み間隔＞より1秒程度増えることがあります。

例　題　ON INTERVAL GOSUB命令の例を参照してください。

関連命令

ON INTERVAL GOSUB	：インターバルタイマ割り込みルーチンを定義します。
INTERVAL ON／OFF／STOP	：インターバルタイマ割り込みを許可、禁止または停止します。

INTERVAL ON/OFF/STOP

インターバル・オン/オフ/ストップ　*interval on/off/stop*　　プログラムの分岐・時計

```
INTERVAL {ON
          OFF
          STOP}
```

機　能　インターバルタイマ割り込みを許可、禁止または停止します。

説　明
- INTERVAL ON
  割り込み動作を許可します。割り込みが発生すると、ON INTERVAL GOSUB命令の定義するインターバルタイマ割り込み処理ルーチンが呼び出されます。割り込みの発生する間隔は、INTERVAL命令で指定した間隔です。
- INTERVAL OFF
  インターバルタイマ割り込みを禁止します。
- INTERVAL STOP
  割り込み動作を一時停止します。命令を実行した後は、割り込みがかかっても、割り込み動作はその時点では行われずに保留され、次にINTERVAL ON命令が実行されたときに、割り込み処理ルーチンの呼び出しが行われます。
- INTERVAL ON／OFF／STOP命令を実行する前に、ON INTERVAL GOSUB命令でインターバルタイマ割り込み処理ルーチンを定義しておく必要があります。

> **注意**　プログラム終了時には、INTERVAL OFF命令を実行してください。

例　題　ON INTERVAL GOSUBを参照してください。

関連命令

ON INTERVAL GOSUB	：インターバルタイマ割り込み処理ルーチンを定義します。
INTERVAL	：インターバルタイマ割り込み間隔を指定します。

I

JIS関数

ジス関数　*jis*　　文字列処理

JIS（文字式）

機　能　日本語文字を、JISコードに変換します。

説　明　<文字式>で指定した文字列の、最初の日本語文字（シフトJISコード）を、JISコードに変換します。

<文字式>……文字列を指定します。

●文字列の最初の文字が日本語文字でない場合は、ASC関数と同様に文字のキャラクタコードを返します。

例　題　日本語文字列をJISコードに変換し、得られたJISコードを16進数に変換して表示します。

```
1000 A$="富士通"
1100 B=JIS(A$)
1200 C$=HEX$(B)
1300 PRINT A$
1400 PRINT C$
1500 END

結果：富士通
      4959
```

関連命令　KNJ$関数：JIS関数とは逆に、JISコードを日本語文字に変換します。
ASC関数　：文字をキャラクタコードに変換します。

KACNV$関数

ケー・エー・コンバート・ダラー関数 *kanji to ANK convert$*　　文字列処理

```
KACNV$（文字式）
```

機　能　文字列中の日本語文字を、対応するANK文字に変換します。

説　明　<文字式>……日本語文字列を含む文字列を指定します。

●日本語文字の英数字、カタカナは、ANK文字の英数字、カタカナに各々変換されます。変換できない文字はそのまま残ります。

例　題　日本語文字を対応するANK文字に変換して、日本語文字とANK文字を表示します。

```
1000 A$="ＦＵＪＩＴＳＵ"
1100 B$=KACNV$(A$)
1200 PRINT A$
1300 PRINT B$
1400 END

結果：ＦＵＪＩＴＳＵ
      FUJITSU
```

関連命令　AKCNV$関数：KACNV$関数とは逆に、ANK文字を日本語文字に変換します。

K

KEXT$関数

ケー・イクストラクト・ダラー関数　*kanji extract$*　文字列処理

KEXT$（文字式，機能）

機　能　文字列の中から、ANK文字列または日本語文字列を取り出します。

説　明　<文字式>で指定した文字列の中から、ANK文字列または日本語文字列を取り出します。

<文字式>……取り出す文字列が含まれている文字列を指定します。

<機能>……取り出す文字列の種類を0または1で指定します。
0：文字列中の全てのANK文字を取り出します。
1：文字列中の全ての日本語文字を取り出します。

例　題　ANK文字と日本語文字が混在する文字列の中から、ANK文字列と日本語文字列に分けて取り出し表示します。

```
1000 A$ ="日本語文字 alphabet Numeric"
1100 B$=KEXT$(A$,0)
1200 C$=KEXT$(A$,1)
1300 PRINT B$
1400 PRINT C$
1500 END

結果：alphabet Numeric
　　　日本語文字
```

KEY

キー　*key*

```
KEY　PFキー番号，文字列
```

機　能　12個のPFキーのうち、PF1キー～PF10キーに文字列を定義します。

説　明　＜PFキー番号＞……キーボード上のPFキーの番号を、1～10の数字で指定します。

＜文字列＞　……最大15バイトまでの文字、またはコントロールキャラクタを指定します。
コントロールキャラクタを指定する場合、CHR$関数を加算記号（+）で接続して指定します。

- KEY命令の実行後、INPUT命令やINKEY$命令を実行してキー入力が可能な状態になったとき、指定したPFキーを押すと、KEY命令で定義した文字列が入力されます。
- INPUT命令やINKEY$命令の実行中は、PFキー割り込み停止（KEY（n）STOP）になっているので、PFキーによる割り込みは発生しません。



KEY命令は、PFキー割り込み禁止（KEY（n）OFF）の状態でだけ有効です。

例　題　PF1キーに「東京」、PF2キーに「名古屋」、PF3キーに「大阪」という文字列を定義します。

```
1000 KEY 1,"東京"
1100 KEY 2,"名古屋"
1200 KEY 3,"大阪"
```

関連命令　KEY LIST：PFキーに定義されている文字列を画面に表示します。
CHR$関数　：キャラクタコードをANK文字に変換します。

KEY LIST

キー・リスト　*key list*


```
KEY LIST
```

機　能　PF1キー～PF10キーに定義されている文字列を、画面に一覧表示します。

説　明　●BASIC起動時には、PF1キー～PF10キー全てに空文字列が設定されています。

関連命令　KEY：PFキーに文字列を定義します。

KEY(n)ON/OFF/STOP

キー・オン/オフ/ストップ　*key(n) on/off/stop*

```
KEY（PFキー番号） { ON
                    OFF
                    STOP }
```

機　能　PFキーからの割り込みを許可、禁止または停止します。

説　明　<PFキー番号>……キーボード上のPFキーの番号を1～10の範囲で指定します。

●KEY(n) ON
PFnキーからの割り込みを可能にします。
命令実行後、PFnキーを押すと、ON KEY（n）GOSUB命令で定義されている割り込み処理ルーチンが呼び出されます。

●KEY(n) OFF
PFnキーからの割り込みを禁止します。

●KEY(n) STOP
PFnキーからの割り込みを一時停止します。
命令実行後、PFnキーを押しても、その時点では、割り込み処理ルーチンの呼び出しは行われず、保留されます。次に、KEY（n）ONを実行すると、割り込み処理ルーチンが呼び出されます。

●KEY(n)ONを実行する前に、ON KEY（n）GOSUB命令によって、PFキー割り込み処理ルーチンを定義しておく必要があります。

●プログラムの実行開始時は、全てのPFキーは、割り込み禁止(OFF)の状態です。

例　題　ON　KEY　GOSUBの例題を参照してください。

関連命令　ON　KEY(n) GOSUB：PFキー割り込み処理ルーチンを定義します。
KEY　　　　　　　　：PFキーに文字列を定義します。

KILL

キル *kill*　　データファイル

```
KILL　ファイルディスクリプタ
```

機　能　ファイルを削除します。

説　明　<ファイルディスクリプタ>で指定されたファイルを削除します。

<ファイルディスクリプタ>……削除するファイルを指定します。ファイルディスクリプタの<オプション>は指定できません。

●削除の指定をするファイルは、クローズ状態でなければなりません。
●BASICエディタでファイルの「削除」を選択したときと同じ結果になります。

例　題　ファイル"EX2"を削除します。

```
KILL "EX2"
```

KINSTR関数

ケー・インストリング関数　*kanji instring*　　　文字列処理

```
KINSTR (〔検索開始位置, 〕文字式 1, 文字式 2 )
```

機　能　文字列を検索して、指定した文字列を見つけ、その位置を文字数で返します。

説　明　＜文字式1＞で指定した文字列から＜文字式2＞で指定した文字列を探し、文字列の先頭位置が＜文字式1＞の先頭から何文字目にあるかを返します。

＜検索開始位置＞……＜文字式1＞で示す文字列の何番目の文字から検索を開始するかを、1～255の数値で指定します。
省略すると、1を指定したとみなされれます。

＜文字式1＞　……検索される文字列を指定します。
文字列の長さは0～255バイトです。

＜文字式2＞　……探し出す文字列を指定します。
文字列の長さは0～255バイトです。

- 日本語文字、ANK文字のどちらも1文字と数えます。
- ＜文字式2＞が見つからなかった場合は、0が返されます。
  ＜文字式1＞が空文字列の場合、および＜検索開始位置＞が＜文字式1＞より大きい場合も、0が返されます。
- ＜文字式2＞が空文字列の場合は、＜検索開始位置＞が返されます。



空白には、半角の空白と全角の空白があります。表示された場合に同じであっても、半角か全角かによって返される値が変わります。

例　題　日本語文字とANK文字からなる文字列から“p”を探して、先頭からの文字数を表示します。

```
1000 A$="日本語文字 alphabet Numeric"
1100 B$="p"
1200 C=KINSTR(A$,B$)
1300 PRINT C
1400 END

結果：9
```

関連命令　INSTR関数：指定された文字列を見つけ、その位置をバイト数で返します。

KLEFT$関数

ケー・レフト・ダラー関数　*kanji left$*　　文字列処理

KLEFT$（文字式, 文字数）

機　能　文字列の左端から、指定した文字数の文字列を取り出します。

説　明　<文字式>で指定した文字列の左端から、<文字数>で指定した長さの文字列を取り出します。

<文字式>……取り出す文字列が含まれている文字列を指定します。

<文字数>……取り出す文字列の長さを指定します。
指定できる範囲は0～255です。
0を指定すると空文字列になります。

- ANK文字、日本語文字ともに1文字と数えます。
- <文字数>が<文字式>で指定する文字列の長さよりも大きい場合は、取り出す文字列は<文字式>で指定した文字列になります。

例　題　指定した文字列の左端から5文字分を取り出し、指定した文字列および取り出した文字列を表示します。

```
1000 A$="日本語文字 alphabet Numeric"
1100 B$=KLEFT$(A$,5)
1200 PRINT A$             '5文字分取り出す
1300 PRINT B$
1400 END

結果：日本語文字 alphabet Numeric
      日本語文字
```

関連命令

KMID$関数　：文字列の中から指定した文字数の文字列を取り出します。

KRIGHT$関数：文字列の右端から、指定した文字数の文字列を取り出します。

LEFT$関数　：文字列の左端から、指定したバイト数の文字列を取り出します。

KLEN関数

ケー・レンクス関数　*kanji length*

文字列処理

KLEN（文字式〔, 機能〕）

機　能　文字列の文字数を返します。

説　明　＜文字式＞で指定した文字列の文字数を返します。

＜文字式＞……日本語文字列を含む文字列を指定します。

＜機能＞　……数える文字の種類を、0～2の数値で指定します。

0: 日本語文字、ANK文字を合わせた文字数

1: ANK文字の文字数

2: 日本語文字の文字数

省略すると、0を指定したとみなされます。

- 日本語文字、ANK文字ともに1文字を1と数えます。また、空白も1文字と数えます。



空白には、半角の空白と全角の空白があります。表示された場合に同じであっても、半角か全角かによって返される値が変わります。

K

例　題　日本語文字とANK文字がそれぞれ何文字含まれているかを表示します。

```
1000 A$="日本語文字 alphabet Numeric"
1100 B=KLEN(A$,0)                        ' 日本語文字, ANK 文字数を与える
1200 C=KLEN(A$,1)                        ' ANK 文字数を与える
1300 D=KLEN(A$,2)                        ' 日本語文字数を与える
1400 PRINT B
1500 PRINT C
1600 PRINT D
1700 END

結果：22
      17
      5
```

関連命令　LEN：文字列のバイト数を返します。

KMID$関数

ケー・ミッド・ダラー関数　*kanji middle$*　　文字列処理

```
KMID$（文字式，文字位置〔，文字数〕）
```

機　能　文字列の中から指定した文字数の文字列を取り出します。

説　明　＜文字式＞で指定した文字列の中の＜文字位置＞から、＜文字数＞で指定した長さの文字列を取り出します。

＜文字式＞　……取り出す文字列が含まれている文字列を指定します。

＜文字位置＞……文字列の先頭から数えて、何文字目から取り出すかを0～255の数値で指定します。

＜文字数＞　……取り出す文字列の長さを0～255の数値で指定します。
省略すると、255を指定したとみなされます。

- ANK文字、日本語文字ともに1文字と数えます。
- 次の場合、結果は空文字列になります。
  - ・＜文字位置＞が＜文字式＞の長さよりも大きい場合。
  - ・＜文字数＞に0を指定した場合。
- ＜文字数＞が＜文字位置＞から最後までの文字列の長さよりも大きい場合は、＜文字位置＞から最後までの文字列が取り出されます。

K

例　題　文字列の7文字目から8文字を取り出し表示します。

```
1000 A$="日本語文字 alphabet Numeric"
1100 B$=KMID$(A$,7,8)
1200 PRINT A$
1300 PRINT B$
1400 END

結果：日本語文字 alphabet Numeric
      alphabet
```

関連命令　KLEFT$関数　：文字列の左端から、指定した文字数の文字列を取り出します。
KRIGHT$関数：文字列の右端から、指定した文字数の文字列を取り出します。
MID$関数　　：文字列の中から指定したバイト数の文字を取り出すか、または文字列の一部に他の文字列を代入します。

KNJ$関数

カンジ・ダラー関数　*kanji$*　　　文字列処理

```
KNJ$（式）
```

機　能　JISコードを、対応する日本語文字（シフトJISコード）に変換します。

説　明　<式>で指定したJISコードを、対応する1文字の日本語文字（シフトJISコード）に変換します。

<式>……JISコードを指定します。

例　題　JISコードを日本語文字に変換して、表示します。

```
1000 A=&H4959
1100 B=&H3B4E
1200 C=&H444C
1300 A$=KNJ$(A)
1400 B$=KNJ$(B)
1500 C$=KNJ$(C)
1600 PRINT A$+B$+C$
1700 END

結果：富士通
```

関連命令　JIS関数　：KNJ$関数とは逆に、日本語文字をJISコードに変換します。
CHR$関数：キャラクタコードをANK文字に変換します。

K

KRIGHT$関数

ケー・ライト・ダラー関数　*kanji right$*

文字列処理

```
KRIGHT$（文字式，文字数）
```

機　能　文字列の右端から、指定した文字数の文字列を取り出します。

説　明　<文字式>で指定した文字列の右端から、<文字数>で指定した長さの文字列を取り出します。

<文字式>……取り出す文字列が含まれている文字列を指定します。

<文字数>……取り出す文字列の長さを指定します。
指定できる範囲は0～255です。

- ANK文字、日本語文字ともに1文字と数えます。
- <文字数>が<文字式>で示される文字列の長さよりも大きい場合、取り出す文字列は<文字式>で指定した文字列になります。
- <文字数>に0を指定すると空文字列になります。

例　題　指定した文字列の右端から7文字分を取り出し、指定した文字列および取り出した文字列を表示します。

```
1000 A$ ="日本語文字 alphabet Numeric"
1100 B$=KRIGHT$(A$,7)                     ' 7 文字分取り出す
1200 PRINT A$
1300 PRINT B$
1400 END

結果：日本語文字  alphabet Numeric
      Numeric
```

関連命令

KLEFT$関数　：文字列の左端から、指定した文字数の文字列を取り出します。
KMID$関数　　：文字列の中から指定した文字数の文字列を取り出します。
RIGHT$関数：文字列の右端から、指定したバイト数の文字列を取り出します。

KTYPE関数

ケー・タイプ関数　*kanji type*

文字列処理

```
KTYPE（文字式, 文字位置）
```

機　能　指定された位置にある文字が、日本語文字かANK文字かを判別します。

説　明　＜文字式＞で指定した文字列の、＜文字位置＞で指定した位置にある文字が、日本語文字かANK文字かを判別し、数値を返します。

＜文字式＞　……日本語文字を含む文字列を指定します。

＜文字位置＞……文字列の先頭から何番目の文字を判別するかを指定します。

- 判別結果は、0または1で返します。
  0: ANK文字
  1: 日本語文字
- ＜文字式＞が空文字列のときは、0が返されます。
- 日本語文字、ANK文字ともに1文字を一つと数えます。また、空白も1文字と数えます。

例　題　文字列内の指定した位置にある文字の種類を、判別して表示します。

```
1000 A$ ="日本語文字 alphabet Numeric"
1100 B=KTYPE(A$,1)                    ' 1 文字目の文字の種類を与える
1200 C=KTYPE(A$,8)                    ' 8 文字目の文字の種類を与える
1300 PRINT B
1400 PRINT C
1500 END

結果：1                               ' 日本語文字
      0                               ' ANK 文字
```

LEFT$関数

レフト・ダラー関数　*left$*

文字列処理

```
LEFT$（文字式，バイト数）
```

機　能　文字列の左端から、指定したバイト数の文字列を取り出します。

説　明　<文字式>で指定した文字列の左端から、<バイト数>で指定した長さの文字列を取り出します。

<文字式>　……文字列を指定します。

<バイト数>……取り出す文字列の長さを0～255の数値で指定します。

- ANK文字は1文字1バイト、日本語文字は1文字2バイトと数えます。
- <バイト数>が<文字式>で示される文字列のバイト数より長い場合は、<文字式>で指定した文字列になります。
- <バイト数>に0を指定すると空文字列になります。

例　題　指定した文字列の左端から7バイト分を取り出して表示します。

```
1000 A$="FUJITSU WORK STATION"
1100 B$=LEFT$(A$,7)              ' 左から7 バイト分取り出す
1200 PRINT B$
1300 END

結果：FUJITSU
```

関連命令　KLEFT$関数：文字列の左端から、指定した文字数の文字列を取り出します。
MID$関数　：文字列の中から指定したバイト数の文字列を取り出します。
RIGHT$関数：文字列の右端から、指定したバイト数の文字列を取り出します。

L

LEN関数

レンクス関数 *length*

文字列処理

LEN（文字式）

機　能　文字列のバイト数を返します。

説　明　<文字式>で指定した文字列の長さを数えて、バイト数で返します。

<文字式>……文字列を指定します。

- 文字列が空文字列の場合は0を返します。
- 日本語文字は1文字2バイト、ANK文字は1文字1バイトで数えます。

例　題　指定した文字列のバイト数を求めて、表示します。

```
1000 A$="FUJITSU F-BASIC386"
1100 A=LEN(A$)
1200 PRINT A
1300 END

結果：18
```

関連命令　KLEN関数：文字列の文字数を返します。

LET

レット　*let*　　　　計算

〔LET〕　変数名＝式

機　能　右辺の式の結果を左辺の変数に代入します。

説　明　＜変数名＞……右辺の＜式＞が数値式の場合は数値変数を指定し、文字式の場合は文字変数を指定します。

＜式＞　　……数値式または文字式を指定します。

例　題　数値式を数値変数に、文字式を文字変数に代入し表示します。

```
1000 LET A=12345
1100 LET B$ ="富士通株式会社"
1200 PRINT A;B$
1300 END

結果：12345　富士通株式会社
```

関連命令　LSET：文字列を左詰めでランダムファイルのバッファに代入します。
RSET：文字列を右詰めでランダムファイルのバッファに代入します。

LINE

ライン *line*　　　　画面表示

```
LINE [ { (wx1, wy1)     } ] - { (wx2, wy2)     } , 論理操作  [, [色]
       { STEP (x1, y1)  }     { STEP (x2, y2)  }

     [ , { [B] [, ラインスタイル]                  } ]
         { BF [, { 中塗り色           } ]         }
         {       { タイルストリング    }           }
```

機　能　2点を結ぶ線または長方形を描きます。

説　明　<wx1, wy1><wx2, wy2> ……描く線の両端、または長方形の対角線の両端をワールド座標値で指定します。<wx1, wy1>を省略すると、最終参照座標（LP）を指定したとみなされます。

<STEP（x1, y1）><STEP（x2, y2）>……描く線の両端、または長方形の対角線の両端を最終参照座標(LP)からの距離（x1, y1）で指定します。LPは、点を指定するごとにその座標に移動していきます。
省略すると、最終参照座標（LP）を指定したとみなされます。

<論理操作>　……色についての操作（PSET、PRESET、AND、OR、XOR、OPAQUEまたはPASTEL）を指定します。
（☞「●グラフィック命令の論理操作」p.60参照）

<色>　……表示する線の色を指定します。
省略すると、直前のCOLOR命令の<前景色>で表示されます。

<B> ……2つの座標を結ぶ線を対角線とする長方形を描きます。

<BF> ……2つの座標を結ぶ線を対角線とする長方形の中を<中塗り色>、または<タイルストリング>で塗りつぶします。

<B><BF>を省略すると、2つの座標を結ぶ直線を描きます。

<ラインスタイル>……BF以外を指定したときに、画面に描く線の種類を指定します。16ドットを単位として、&H0〜&HFFFFまでの値で指定します。
線は<ラインスタイル>の“1”のビットに対応するドットが表示され16ドットごとに繰り返します。(指定のしかたについては、例題1を参照してください。)

<中塗り色> ……BFを指定したときに、長方形の内部を塗りつぶす色を指定します。

<タイルストリング>……BFを指定したときに、長方形の内部を塗りつぶすタイルパターンを文字式で指定します。
(☞「●パターンの指定」p.61参照)
<中塗り色>および<タイルストリング>を省略すると、<色>で長方形の内部を塗りつぶします。

●LINE命令を実行後、最終参照座標(LP)は、<wx2, wy2>になります。

L

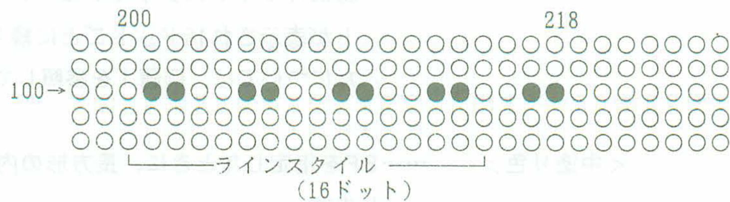
例題 1　ラインスタイルを使用して破線を描きます。

```
LINE (200,100)-(218,100), PSET,7,, &H6666
```

&H6666は次のような16ビットです.

0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0

命令を実行すると，次のようなラインが描かれます.



例題 2　四角形を表示します。

```
1000  CLS
1100  LINE (60,20)-(110,70),PSET,7,B          ' 箱の表示をする.
1200  LINE (120,20)-(170,70),PSET,7,BF        ' 箱の中を塗る.
1300  LINE(180,20)-(230,70),PSET,7,B,&H0707   ' 箱を点線で表示する.
1400  END
```

関連命令　CONNECT：複数の点を結ぶ線または多角形を描きます。

LINE INPUT

ライン・インプット *line input*

データ入力

```
LINE INPUT ["プロンプトメッセージ" {,|;}] 文字変数
```

機　能　キーボードから１行のデータを読み込み、変数に代入します。

説　明　キーボードから１行を読み込み、区切ることなく＜文字変数＞に代入します。入力時には、＜プロンプトメッセージ＞が表示され、入力待ち状態になります。

＜プロンプトメッセージ＞……入力を促すメッセージを指定します。
疑問符（?）は表示されません。
省略すると、何も表示されず入力待ち状態になります。

＜文字変数名＞……データが代入される文字変数を指定します。

- １行とは、CRコードが現れるまでの、BREAKキーを除く全ての文字です。CR、BREAK以外のすべての文字コードが代入されます。
- 読み込める文字数は、最大255バイトです。
- 入力されたデータにコンマ（,）やコロン(:)が存在しても、区切り文字として判断せず、データとして代入されます。
- データ入力中に、BREAKキーまたはCTRL+Cキーが押されると、プログラムの実行を中断し、BASICエディタに戻ります。中断後に実行継続（Continue）が指示されると、LINE INPUT命令から実行を再開します。
- CRコードが現れるまで、命令は終了しません。

例　題　文字変数にデータを代入し、入力結果を表示します。

```
1000 PRINT "ＬＩＮＥ　ＩＮＰＵＴで，変数へ代入"
1100 LINE INPUT "B$:";B$     '区切り記号もデータとして読み込む
1200 PRINT "B$=";B$
1300 END

結果：ＬＩＮＥ　ＩＮＰＵＴで，変数へ代入
      B$:1,2,3
      B$=1,2,3
```

関連命令　INPUT：キーボードからデータを読み込みます。

LINE INPUT#

ライン・インプット・シャープ　*line input#*

データ入力・データファイル

```
LINE　INPUT#　ファイル番号，文字変数
```

機　能　ファイルからデータを１行読み込み、変数に代入します。

説　明　＜ファイル番号＞……オープン時にファイルに割り当てたファイル番号を指定します。

＜文字変数＞　……読み込んだデータを代入する変数を指定します。
読み込める文字数は、1～255バイトの範囲です。

- ＜ファイル番号＞で指定するファイルは、LINE INPUT#命令が実行される前に、入力モード（I）でオープンされている必要があります。
- １行とは、CRコードが現われるまでの全ての文字です。
- コンマ（,）は、データの区切り文字とはみなさず、データとして読み込みます。CRコード以外のコントロールコードは、全て読み飛ばします。
- CRコードが現れるまで、命令は終了しません。

例　題　ファイル“DATA3DAT”からデータを読み込み、表示します。
LINE INPUT#では、コンマ（,）もデータとみなされますから、“1,2,3” は１行のデータとして入力されます。

```
1000 OPEN "O",#1,"DATA3DAT"
1100 PRINT #1,"1,2,3"
1200 CLOSE #1
1300 OPEN "I",#1,"DATA3DAT"
1400 WHILE EOF(1)<>-1
1500   LINE INPUT #1,A$:PRINT A$     'コンマもデータとして読み込む
1600 WEND
1700 CLOSE #1:END

結果：1,2,3
```

関連命令　INPUT#：ファイルからデータを読み込み、変数に代入します。

L

LIST（省略形L.）

リスト　*list*　　　　　　　　　　　　　　　　　　　　　　　　コマンド

1

```
LIST [{行番号1  } ] [{,} ] [{行番号 2  } ]
     [{ラベル名 1} ] [{-} ] [{ラベル名 2} ]
```

機　能　メモリ内のプログラムを、実行画面に出力します。

説　明　＜行番号＞または＜ラベル名＞で指定した範囲を出力します。

＜行番号1＞ ＜行番号2＞　……出力する範囲を行番号で指定します。

＜ラベル名1＞ ＜ラベル名2＞……出力する範囲をラベル名で指定します。

●LIST　行番号1－行番号 2

＜行番号1＞または＜ラベル名1＞と、＜行番号2＞または＜ラベル名2＞をハイフン（－）またはコンマ（,）でつないで記述した場合は、その 2 つの行番号またはラベル名にはさまれた行を出力します。

●LIST　行番号1－

＜行番号1＞または＜ラベル名1＞の後ろに、ハイフン（－）またはコンマ（,）のみを記述した場合は、＜行番号1＞または＜ラベル名1＞以降の全ての行を出力します。

●LIST　行番号 1

＜行番号1＞または＜ラベル名1＞のみを記述した場合は、指定した行のみを出力します。

●LIST　－行番号 2

ハイフン（－）またはコンマ（,）に続けて＜行番号2＞または＜ラベル名2＞を記述した場合は、プログラムの先頭の行から＜行番号2＞または＜ラベル名2＞までの行を出力します。

●LIST　－

ハイフン（－）またはコンマ（,）のみを記述した場合、またはLISTのみを記述した場合は、プログラムの全ての行を出力します。

●LIST.
プログラム実行中にエラーが発生した直後、または編集時に、行番号の代わりにピリオドを指定したLIST命令を一行実行すると、エラーの発生した行または最後に編集した行を画面に出力します。

●プログラム中にLIST命令があると、LIST命令を実行後、プログラムを終了します。エディタの一行実行でLIST命令を実行した場合と同じです。

コンパイラ ●コンパイラでは、LIST命令は使用できません。LIST命令を含むプログラムをコンパイルするとエラーになります。

例　題　行番号をいくつか指定して、画面とプリンタに出力します。

```
LIST              ' 全ての行が画面に表示されます
LIST 10,300       ' 行番号10から300 までの行が表示され
                    ます
LIST.             ' エラーの発生した行または編集行が表
                    示されます
```

2 LIST ファイルディスクリプタ 〔, {行番号1 / ラベル名1}〕〔{, / -}〔{行番号2 / ラベル名2}〕〕

機　能　メモリ内のプログラムをファイルに出力します。

説　明　<ファイルディスクリプタ>……出力するファイルのファイルディスクリプタを指定します。

（☞「●ファイルディスクリプタの形式」p.31参照）

<行番号1> <行番号2> <ラベル名1> <ラベル名2>……出力する範囲を指定します。省略した場合の動作は、形式1と同じです。

●ファイルディスクリプタのデバイス名にディスクを指定すると、SAVE命令でアスキー形式を指定したのと同じファイルを出力します。

コンパイラ　●コンパイラでは、LIST命令は使用できません。LIST命令を含むプログラムをコンパイルするとエラーになります。

例　題　指定したプログラムの行番号をファイルに出力します。

```
LIST "LPT0:",10,300      ' プリンタに行番号10から300 までの行
                           が出力されます
```

関連命令　LLIST：メモリ内のプログラムをプリンタに出力します。
SAVE：メモリ内のプログラムをファイルに格納します。

LLIST

エル・リスト　*list to line printer*　　コマンド

```
LLIST [{行番号1 }] [{,}] [{行番号2 }]
       {ラベル名1}    {-}    {ラベル名2}
```

機　能　メモリ内のプログラムを、プリンタに出力します。

説　明　出力結果は、LIST　"LPT0:"を実行したときと全く同じになります。

<行番号1>　<行番号2>　……出力する範囲を行番号で指定します。

<ラベル名1>　<ラベル名2>……出力する範囲をラベル名で指定します。

● LLIST　行番号1−行番号2
<行番号1>または<ラベル名1>と、<行番号2>または<ラベル名2>を、ハイフン（−）またはコンマ（,）でつないで記述した場合は、その2つの行番号またはラベル名にはさまれた行を出力します。

● LLIST　行番号1−
<行番号1>または<ラベル名1>の後ろに、ハイフン（−）またはコンマ（,）のみを記述した場合は、<行番号1>または<ラベル名1>以降の全ての行を出力します。

● LLIST　行番号1
<行番号1>または<ラベル名1>のみを記述した場合は、指定した行のみを出力します。

● LLIST　−行番号2
ハイフン（−）またはコンマ（,）に続けて<行番号2>または<ラベル名2>を記述した場合は、プログラムの先頭の行から<行番号2>または<ラベル名2>までの行を出力します。

● LLIST　−
ハイフン（−）またはコンマ（,）のみを記述した場合、またはLLISTのみを記述した場合は、プログラムの全ての行を出力します。

● LLIST.
プログラム実行中にエラーが発生した直後、または編集時に、行番号の代わりにピリオドを指定したLLIST命令を一行実行すると、エラーの発生した行、または最後に編集した行を出力します。

● プログラムの中でLLIST命令を実行した場合、実行後、プログラムを終了し、BASICエディタに戻ります。

コンパイラ ● コンパイラでは、LLIST命令は使用できません。LLIST命令を含むプログラムをコンパイルするとエラーになります。

例　題　いくつかの行を指定して、プリンタに出力します。

LLIST	すべての行が印刷されます.
LLIST 10,300	行番号10から300までの行が印刷されます.
LLIST ,100	プログラムの最初から行番号100までの行が印刷されます.

関連命令　LIST：メモリ内にあるプログラムの全部または一部を、画面に表示します。

LOAD〔省略形LO.〕

ロード *load*　　コマンド

```
LOAD ファイルディスクリプタ 〔,R〕
```

機　能　ファイル上のプログラムをメモリに読み込みます。

説　明　<ファイルディスクリプタ>で指定したファイルから、プログラムを読み込みます。

<ファイルディスクリプタ>……読み込むプログラムが格納されたファイルを指定します。
（☞「●ファイルディスクリプタの形式」p.31参照）

<R>……オープンされているファイルのクローズ処理を行わないで、プログラムを読み込んで実行します。省略すると、読み込んだプログラムを実行しません。この場合オープンされている全てのファイルをクローズします。

●プログラムの中でLOAD命令を実行した場合、LOAD命令の実行後、直ちにプログラムを終了し、BASICエディタに戻ります。エディタの画面で一行実行をした場合と同じです。

●BASICエディタでファイルの「読み込み」を指定した場合と同じ結果になります。

コンパイラ　●コンパイラでは、LOAD命令は使用できません。LOAD命令を含むプログラムをコンパイルするとエラーになります。

関連命令　SAVE　：メモリ内のプログラムをファイルに格納します。
MERGE：メモリ内のプログラムと指定されたファイルのプログラムを、メモリ上で混ぜ合わせます。

L

LOAD@

ロード・アットマーク　*load@*

音楽/音声・画面表示

1　LOAD@　〔ファイルディスクリプタ〕

ただし、ファイル名の拡張子は.PMBまたは.FMBです。

機　能　PCM音源、またはFM音源の音色データファイル（".PMBファイル"、または".FMBファイル"）を、PCM音源の専用メモリ(waveRAM)またはFM音源の専用メモリに読み込みます。

説　明　<ファイルディスクリプタ>……TownsSOUNDなどで出力した、拡張子が、".PMB"のPCM音源の音色データファイル、または拡張子が".FMB"のFM音源の音色データファイルを指定します。
省略すると、システム標準のPCM音色データおよびFM音色データが、PCM音源の専用メモリ（waveRAM）およびFM音源の専用メモリに読み込まれます。

- LOAD@命令は、FM音色（@1～@128）またはPCM音色（@1～@32）または両方を全部入れ替えます。
- PCMPLAY命令を実行すると、waveRAMにあるPCM音色データは使用できなくなります。このあと、PLAY命令でPCM音色データを利用するには、ファイルディスクリプタを省略したLOAD@命令を実行して音色データを復旧する必要があります。

関連命令　PLAY　　　：音楽の演奏を行います。
PCMPLAY：PCM音声データをPCM音源に送って発声させます。
SAVE@　　：音色データまたはイメージデータをファイルに格納します。

2

```
LOAD@ ファイルディスクリプタ〔,( x, y )〕
```

ただし、ファイル名の拡張子は．TIF又は、．JPGです。

機　能　イメージファイル（絵のファイル）を読み込み、グラフィック画面に表示します。

説　明　<ファイルディスクリプタ>……TownsPAINTなどで出力した、拡張子が".TIF"のTIFFファイル、又は拡張子が".JPG"のJPEGファイルを指定します。

<(x, y)>………………………イメージを表示する画面上の領域の左上隅を、オリジナルスクリーン座標で指定します。
省略すると、(0, 0) を指定したとみなされ、画面の左上隅から表示します。

- モノクロのイメージデータの場合は、COLOR命令の<前景色>で表示されます。カラーのイメージデータの場合は、画面モードがイメージファイル中の画面モードと一致していないと表示されません。
- パレット情報をもったイメージデータを読み込むと、この情報によってパレットの色が設定されます。
- 「圧縮情報」を指定して格納したイメージデータも読み込むことができます。（V1.1のBASICでは読み込めません。）
- 画面モードが3万2千色モードの時、JPEGファイルも読み込むことができます。
- JPEGファイルを読み込む時は、CLEAR命令でDLL領域を確保してください。

関連命令　PUT@　：GET命令（形式1）で配列に読み込まれたドットパターンを画面に表示します。
PUT@A：GET命令（形式2）で配列に読み込まれたグラフィックデータを画面に表示します。

3 LOAD@ ファイルディスクリプタ，配列名 [，{0 / 1}]

ただし、ファイル名の拡張子は.EUPです。

機　能　EUPファイルを配列に読み込み、音色ファイルをセットします。

説　明　<ファイルディスクリプタ>……拡張子が".EUP"のファイルを指定します。

<配列名>　……………………EUPファイルのデータが入るだけの大きさを持った配列の変数名を指定します。文字型の配列は指定できません。

<0>、<1>…………………EUPファイルのヘッダに書かれている音色ファイルを読み込むかどうかを指定します。

0：音色ファイルを読み込む。

1：音色ファイルを読み込まない。(すでにセットされている音色を使う。)

省略したときは、0を指定したとみなされます。

●V1.1のBASICでは、EUPファイルを読み込めません。

関連命令　PLAY@：EUPファイルを演奏します。

4

LOAD@　ファイルディスクリプタ，配列名

ただし、ファイル名の拡張子は、.EUP以外です。

機　能　データファイルを、配列に読み込みます。

説　明　<ファイルディスクリプタ>……拡張子が、".EUP"以外のファイルを指定します。

<配列名>　……………………データファイルが入るだけの、十分な大きさを持った配列の変数名を指定します。文字型の配列は指定できません。

関連命令

PCMPLAY：配列に読み込まれたPCM音色データを発声させます。

VOICE SET：配列に読み込まれたPCM音色データをPCM音源の専用メモリにセットします。

SAVE@：音色データまたはイメージデータをファイルに格納します。

PCMREC：マイクからの音声をサンプリングして、そのデータを配列に入れます。

LOADM

ロード・エム　*load machine program*　　機械語プログラム

```
LOADM ファイルディスクリプタ，オフセット
```

機　能　プロシジャをメモリに読み込みます。

説　明　指定したファイルからCLEAR命令で確保したプロシジャ領域にプロシジャを読み込みます。

＜ファイルディスクリプタ＞……読み込もうとするプロシジャ（実行形式）のファイル名を指定します。
".REX"ファイルを指定します。
これ以外のファイルを指定すると、エラーになります。

＜オフセット＞……読み込み開始アドレスを、プロシジャ領域の先頭からのアドレスで指定します。
ロング型整数で指定します。

●LOADM命令は、以下のように処理を行います。

・ファイルディスクリプタで指定された実行形式ファイルを、読み込み開始アドレスに読み込みます。

・読み込んだ実行形式ファイルに対して、自動的にリロケーション（フィックスアップ）を行います。リロケーションとは、機械語での絶対番地へのジャンプ命令や、コール命令などのアドレス問題を解決するものです。

例　題　LOADM "b:procdr.REX",0&

関連命令　CLEAR：プロシジャ領域の大きさを設定します。

LOC

エル・オー・シー　*location counter of diskfile*　　データファイル

```
LOC（ファイル番号）
```

機　能　ランダムファイルで、次にGETまたはPUTされるレコード番号を返します。

説　明　<ファイル番号>……オープン時に割り当てたランダムファイルのファイル番号を指定します。

- <ファイル番号>で指定したファイルは、ランダム入出力モード（R）で既にオープンされている必要があります。
- 返される値は、直前にGETまたはPUTされたレコードの次のレコードのレコード番号です。
  <レコード番号>を省略したGETまたはPUT命令を実行すると、このレコードがGETまたはPUTされます。

例　題　次に読み書きされるレコード番号を変数Aに返します。

```
1000 OPEN"R",#1,"SAMPLE"
1100 FIELD #1,2 AS A$,4 AS B$,8 AS C$
1200 GET #1,1
1300 A=LOC(1)                        ' 次のレコード番号(2) を与える
1400 PRINT A
1500 CLOSE #1
1600 END

結果：2
```

関連命令　GET：ランダムファイルから、データをバッファに読み込みます。
PUT：バッファの内容を、指定されたランダムファイルに出力します。
LOF：ランダムファイルの最大レコード番号を返します。

L

LOCATE〔省略形LOC.〕

ロケート *locate* 　　画面表示

```
LOCATE 〔水平位置〕〔,〔垂直位置〕〔, カーソルスイッチ〕〕
```

機　能　テキスト画面上の指定した位置に、カーソルを移動します。

説　明　＜水平位置＞、＜垂直位置＞で指定したキャラクタ座標に、カーソルを移動します。

＜水平位置＞＜垂直位置＞……カーソルが移動する位置を、キャラクタ座標値で指定します。
省略すると、カーソルは移動しません。

＜カーソルスイッチ＞……カーソルを表示するか否かを、0または1の数値で指定します。
0：カーソルを画面上に表示しません。
1：カーソルを画面上に表示します。
省略すると、現在の状態のままになります。

- ＜カーソルスイッチ＞に1を指定した場合、INPUT命令またはLINE INPUT命令が実行されて入力待ちになったときに、カーソルが表示されます。
- CONSOLE命令でPFキー表示を指定した場合、その行にカーソル位置を設定することはできません。

例　題　FOR～NEXTループでカーソルを移動させながら、画面表示を行います。

```
1000 WIDTH 80
1100 X=0
1200 FOR Y=0 TO 10
1300  LOCATE X,Y :PRINT "FUJITSU"  ' カーソル位置に文字を表示
1400  FOR T=1 TO 1000 :NEXT T
1500 X=X+2 :NEXT Y
1600 END

結果：FUJITSU
        FUJITSU
          FUJITSU
            FUJITSU
              FUJITSU
                FUJITSU
                  FUJITSU
                    FUJITSU
                      FUJITSU
                        FUJITSU
                          FUJITSU
```


関連命令

POS関数	：カーソルまたはプリンタバッファの水平位置を返します。
CSRLIN関数	：画面上のカーソルの垂直位置を返します。
PRINT	：画面のカーソル位置に式の結果を表示します。
CONSOLE	：テキスト画面のスクロールウィンドウを設定します。
INPUT	：キーボードから入力したデータを変数に読み込みます。
LINE INPUT	：キーボードから入力した１行のデータを、区切ることなく変数に読み込みます。

LOF関数

エル・オー・エフ関数 *length of diskfile*　　データファイル

LOF（ファイル番号）

機　能　入力バッファの中の文字数またはランダムファイルの最大レコード番号を返します。

説　明　＜ファイル番号＞で指定したRS-232Cインタフェースポートの入力バッファに入っている文字数を返します。または、＜ファイル番号＞で指定したランダムファイルの最大レコード番号を返します。

＜ファイル番号＞……オープン時にRS-232Cまたはランダムファイルに割り当てたファイル番号を指定します。

●＜ファイル番号＞に上記以外のファイルを指定することはできません。

例　題　ランダムファイル“DATA1DAT”のデータを読み込み、最大レコード番号が0のとき、または入力されたレコード番号がLOFで得られた最大レコード番号より大きいときは、処理を終了します。

```
1000 OPEN"R",#1,"DATA1DAT"
1100 FIELD#1,20 AS NAM$,12 AS TEL$
1150 IF LOF(1)=0 THEN 1900          ' DATA1DATにデータがあるか
1200 INPUT "レコード番号は";NO
1300 IF LOF(1)<NO THEN 2000         ' 指定レコードにデータがあるか
1400 GET#1,NO
1600 PRINT NAM$; TEL$
1700 GOTO 1200
1900 PRINT"DATA1DAT にはデータが登録されていません. ":CLOSE#1:END
2000 PRINT" レコードにはデータが記録されていません. ":CLOSE#1:END

結果：レコード番号は？1
      警察                110
      レコード番号は？2
      消防                119
      レコード番号は？3
      レコードにはデータが記録されていません.
```

関連命令　EOF関数：ファイルの終わりを検出します。

LOG関数

ログ関数　*logarithm*　　計算

LOG（式）

機　能　自然対数を返します。

説　明　<式>……正の実数を指定します。

●式の型が、倍精度の場合は倍精度の、単精度の場合は単精度の結果を返します。

例　題　入力した倍精度型の数値の、自然対数を求めて表示します。

```
1000 INPUT A
1100 B=LOG(A)
1200 PRINT B
1300 INPUT A#
1400 B#=LOG(A#)
1500 PRINT B#
1600 END

結果：? 20.236
　　　 3.00746
　　　 ? 22.312954
　　　 3.10516740667179
```

関連命令　EXP関数：ｅを底とする指数関数の値を返します。

L

LPOS関数

ラインポジション関数 *line position*

印刷

```
LPOS (式)
```

機　能　プリンタバッファ内の、水平文字位置を返します。

説　明　<式>……式はダミーで意味を持ちませんが、形式上記述します。

例　題　アルファベットのA～Zを、順番にプリンタ装置に印字します。その際、各文字の水平位置を画面に表示します。

```
1000  FOR I=&H41 TO &H5A
1100    LPRINT CHR$(I);
1200    IF  I=&H4E THEN PRINT
1300    PRINT USING "###";LPOS(0);
1400  NEXT I
1500  END
表示結果：  1  2  3  4  5  6  7  8  9 10 11 12 13
           14 15 16 17 18 19 20 21 22 23 24 25 26
印刷結果：  ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

関連命令　POS関数：カーソルまたはプリンタバッファ内の水平文字位置を返します。

LPRINT

エル・プリント　*line print out*

印刷

```
LPRINT 〔式〔{ , | ; }〔式〕〕…〕
```

機　能　式の結果をプリンタに出力します。

説　明　<式>……出力する数値または文字列を指定します。
<式>を全て省略すると、改行のみを行います。

- セミコロン（;）の後の式の結果は、前の式の結果に続いて出力されます。
  式の結果が数値の場合は、数値の前後に1個ずつの空白が取られます。前の空白は符号を印字するための領域で、数値が負のときはマイナス（−）を、正のときは空白が入ります。
- コンマ（,）の後の式の結果は、前の式の結果が表示されているフィールドの次のフィールドの先頭から出力されます。
  式の結果が2つ以上のフィールドにまたがる場合、後の結果は前の結果のフィールドの次のフィールドの先頭から出力されます。



フィールド
1行は半角14文字で区切られています。この単位をフィールドといいます。

- 行の最後の記述が、コンマ（,）またはセミコロン（;）で終わっている場合、次のLPRINT命令による表示が、同じ行に出力されます。コンマ（,）または　セミコロン（;）で終わっていない場合は、自動的に改行します。
- 出力される<式>の長さが用紙の幅より長い場合は、行の終わりまで出力した後、残りを改行後に印字します。
- 現在のプリンタヘッドの位置から行の終わりまでの間に、出力される<式>の長さを確保できない場合には、改行してから出力します。
  ただしWIDTH"LPT0：",0が指定されているときは、改行しないで続けて出

力します。

例　題　　変数の値をプリンタに出力します。

```
1000  A$="1234567890"
1100  B$="ABCDEFG"
1200  C$="HIJKLMN"
1300  LPRINT A$                 ' 印刷後，改行する．
1400  LPRINT B$;C$              ' B$の直後から，C$を印刷する．
1500  END
印刷結果:1234567890
         ABCDEFGHIJKLMN
```

関連命令　PRINT　：式の結果を画面に表示します。
PRINT#：式の結果をファイルに出力します。
WRITE#：式の結果をファイルに出力します。
WRITE　：式の結果を画面に表示します。

LPRINT USING

エル・プリント・ユージング　*line print out using*　　印刷

LPRINT USING　書式文字列；〔式〕〔{ , / ; }〔式〕〕…

機　能　文字または数値を指定した書式でプリンタに出力します。

説　明　<書式文字列>……文字や数値の出力形式を指定します。

（「書式制御文字」p.72参照）

<式>　　……出力する文字列または数値を指定します。

例　題　値や文字を書式指定して表示します。

```
1000 A=12345
1100 B$="ABCDEFGHIJKLMN"
1200 C$="OPQRSTU"
1300 LPRINT USING "#####.##";A
1400 LPRINT USING "&             & &      &";B$;C$
1500 END

結果：12345.00
      ABCDEFGHIJKLMN   OPQRSTU
```

関連命令　PRINT USING：文字または数値を、指定した書式で画面に出力します。

LSET/RSET

エル・セット/アール・セット *left set/right set*

計算・データファイル

```
{LSET}
{RSET} 文字変数＝文字式
```

機　能　左詰めまたは右詰めで、ランダムファイルバッファのフィールドに文字列を代入します。

説　明　<文字変数>に<式>で指定した文字列の値を代入します。

<文字変数>……FIELD命令により、ランダムファイルのバッファに割り当てられた変数を指定します。

<文字式>　……文字列を指定します。

- LSET　文字列をフィールドに左詰めで代入します。残りの部分には、空白が詰められます。
- RSET　文字列をフィールドに右詰めで代入します。残りの部分には、空白が詰められます。
- この命令を実行する前に、FIELD命令でランダムファイルバッファをフィールド分割しておく必要があります。
- <文字式>が<文字変数>より長い場合は、右側の残った文字列が捨てられます。
- 数値を<文字変数>に代入するためには、MKI$、MKL$、MKS$またはMKD$関数のいずれかで、数値を文字列に変換しておく必要があります。

例　題　ランダムファイルのバッファに割り当てた変数に文字列を代入し、ランダムファイルに出力します。更に、出力したレコードをランダムファイルからバッファに読み込み、画面に表示します。

```
1000 OPEN "R",#1,"DATA4DAT"
1100 FIELD #1,10 AS A$,10 AS B$,10 AS C$
1200 LSET A$="AAAAA"
1300 RSET B$="BBBBB"
1400 LSET C$="CCCCCCCCCCCCCCCCCCCCCCCCCCCC"
1500 PUT#1,1
1600 GET#1,1
1700 PRINT A$ : PRINT B$ : PRINT C$
1800 CLOSE#1
1900 END

結果 AAAAA
          BBBBB
     CCCCCCCCCC
```

関連命令　FIELD：ランダムファイルのバッファを分割し、変数の領域を割り当てます。

MAP関数

マップ関数　*map*

画面表示

MAP（座標値，機能）

機　能　ワールド座標をスクリーン座標に、またはスクリーン座標をワールド座標に変換します。

説　明　＜座標値＞で指定したワールド座標値またはスクリーン座標値を、＜機能＞の指定にしたがって、対応するスクリーン座標値またはワールド座標値に変換します。

＜座標値＞……ワールド座標値またはスクリーン座標値を指定します。
ワールド座標値は、以下の範囲で指定します。
x 座標：−3.40282E+38～+3.40282E+38
y 座標：−3.40282E+38～+3.40282E+38

＜機能＞……変換の種類を0～3の数値で指定します。以下の（wx, wy）はワールド座標、（sx, sy）はスクリーン座標を示します。

0 ：wx→sx
＜座標値＞で指定したワールド座標の x 座標値が、対応するスクリーン座標の x 座標値に変換されます。

1 ：wy→sy
＜座標値＞で指定したワールド座標の y 座標値が、対応するスクリーン座標の y 座標値に変換されます。

2 ：sx→wx
＜座標値＞で指定したスクリーン座標の x 座標値が、対応するワールド座標の x 座標値に変換されます。

3 ：sy→wy
＜座標値＞で指定したスクリーン座標の y 座標値が、対応するワールド座標の y 座標値に変換されます。

● ＜座標値＞が、ウィンドウまたはビューポートの範囲外のときも、範囲内の座標と同様に、ウィンドウとビューポートの比率にしたがって値を変換し、その値を

返します。

例　題　スクリーン座標値を、ワールド座標値に変換して表示します。

```
1000  CLS
1100  WINDOW(0,0)-(319,199)
1200  VIEW(0,0)-(639,399)
1300  SX=640 : SY=400
1400  WX=MAP(SX,2)
1500  WY=MAP(SY,3)
1600  PRINT"スクリーン座標 SX= " ; SX
1700  PRINT "スクリーン座標 SY= " ; SY
1800  PRINT "ワールド座標   WX= " ; WX
1900  PRINT "ワールド座標   WY= " ; WY
2000  END
```

関連命令
WINDOW　：ワールド座標内のウィンドウの範囲を指定します。
VIEW　：オリジナルスクリーン座標内のビューポートの範囲を指定します。
WINDOW関数：現在ウィンドウに設定されている値をワールド座標値で返します。
VIEW関数　：現在ビューポートが設定されている位置をオリジナルスクリーン座標で返します。

M

MERGE〔省略形ME.〕

マージ　*merge*　　　　コマンド

```
MERGE　ファイルディスクリプタ〔, R〕
```

機　能　メモリにあるプログラムと、指定されたファイルのプログラムをメモリ上で混ぜ合わせます。

説　明　<ファイルディスクリプタ>……プログラムが格納されたファイルを指定します。
（☞「●ファイルディスクリプタの形式」 p.31参照）

<R>……混ぜ合わせた後に、プログラムの先頭から実行を開始する場合に指定します。
省略すると、混ぜ合わせだけを行います。

- ２つのプログラムを１つにします。
両方に同じ行番号があるとき、メモリ上の行は対応するファイル上の行に置き換えられます。置き換えられないようにするには、あらかじめ２つのプログラムの行番号を分離しておく必要があります。
- プログラムはアスキー形式でセーブされていなければなりません。バイナリ形式のファイルを指定すると、エラーになります。

コンパイラ
- コンパイラでは、MERGE命令は使用できません。MERGE命令を含むプログラムをコンパイラするとエラーになります。

例　題　メモリにあるプログラム（PROG1）と、ファイル上のプログラム（PROG2）を結合します。

```
100  INPUT A                          'メモリ上のプログラム
200  PRINT A
1100 INPUT X

MERGE "1:PROGA.EXE"

結果：100 INPUT A                    '結合されたプログラム
      200 PRINT A
      1000 INPUT A$
      1100 INPUT X
      1200 PRINT A$
      1300 PRINT A$
      1400 END
```

関連命令　LOAD：ファイル上のプログラムをメモリに読み込みます。

MID$関数

ミッド・ダラー関数　*middle$*　　　文字列処理

1　MID$（文字式，文字位置〔，バイト数〕）

機　能　文字列の中から、指定したバイト数の文字列を取り出します。

説　明　<文字式>で指定した文字列の中の<文字位置>から、<バイト数>で指定した長さの文字列を取り出します。

<文字式>　……取り出す文字列が含まれている文字列を指定します。

<文字位置>……文字列の先頭から数えて、何バイト目から取り出すかを1～255の範囲内で指定します。

<バイト数>……取り出す文字列の長さを0～255の範囲内で指定します。省略すると、255を指定したとみなされます。

- ANK文字は1文字1バイト、日本語文字は1文字2バイトと数えます。
- 次の場合、結果は空文字列になります。
  - ・<文字位置>が<文字列>より大きい場合。
  - ・<バイト数>に0を指定した場合。
- <バイト数>が<文字位置>から最後までの文字列の長さよりも大きい場合、取り出す文字列は<文字位置>から最後までの文字列になります。

M

例　題　文字列の９文字目から４バイトを取り出し表示します。

```
1000 A$="FUJITSU F-BASIC/386"
1100 B$=MID$(A$,9,4)
1200 PRINT B$
1300 END

結果：F-BA
```

2　MID$（文字変数名，文字位置〔，バイト数〕）＝文字式

機　能　文字列の一部に他の文字列を代入します。

説　明　＜文字変数名＞で指定した文字列の中の＜文字位置＞から、＜バイト数＞で指定した長さの文字列に＜文字式＞で指定した文字列を代入します。

＜文字変数名＞……代入する元の文字変数を指定します。

＜文字位置＞　……元の文字列の先頭から数えて、何バイト目から代入するかを1～255の範囲内で指定します。

＜バイト数＞　……何バイトの文字列を代入するかを0～255の数値で指定します。省略すると、255を指定したとみなされます。

＜文字式＞　　……代入する文字列を指定します。

- ANK文字は1文字1バイト、日本語文字は1文字2バイトと数えます。
- ＜バイト数＞が＜文字式＞で示される文字列の長さよりも大きい場合は、＜文字式＞の大きさを指定したとみなされます。
- ＜バイト数＞が＜文字式＞で示される文字列の長さよりも大きい場合は、＜文字式＞の大きさを指定したとみなされます。
- ＜バイト数＞に0を指定すると、代入は行われません。

関連命令　KMID$　：文字列の中から指定した文字数の文字列を取り出します。
LEFT$　：文字列の左端から、指定したバイト数分の文字列を取り出します。
RIGHT$：文字列の右端から、指定したバイト数分の文字列を取り出します。

MKI$/MKL$/MKS$/MKD$関数

エム・ケー・アイ・ダラー/エム・ケー・エル・ダラー/ *make integer$/make long$/*
エム・ケー・エス・ダラー/エム・ケー・ディ・ダラー関数 *make single$/make double$*

文字列処理・データファイル

MKI$(整数)
MKL$(ロング型整数)
MKS$(単精度型実数)
MKD$(倍精度型実数)

機　能　数値を文字列に変換します。

説　明　数値データを文字列として扱えるように型変換します。
ランダムファイルに数値データを書き込む場合などに使います。

MKI$　：整数を2バイトの文字列に変換します。
MKL$　：ロング型整数を4バイトの文字列に変換します。
MKS$　：単精度型実数を4バイトの文字列に変換します。
MKD$　：倍精度型実数を8バイトの文字列に変換します。

●ランダムファイルでは文字型のデータだけを取り扱います。そのため、数値データを書き込みたい場合は、数値データをいったん文字列に変換してから書き込みを行います。MKI$、MKL$、MKS$およびMKD$は、その変換を、内部表現形式の数値をそのまま文字列として扱うことによって行います。

M

例　題　入力した値を文字型に変換し、ランダムファイルに書き込みます。書き込んだデータを読み出して数値に変換し表示します。

```
1000 OPEN "R",#1,"SAMPLE"
1100 FIELD #1,4 AS B$
1200 INPUT "B&=";B&
1300 LSET B$=MKL$(B&)
1400 PUT #1,1
1500 GET #1,1
1600 B&=CVL(B$)
1700 PRINT B&
1800 CLOSE #1
1900 END

結果：B&=? 327.68
       328
```

```
1000 OPEN "R",#1,"SAMPLE"
1100 FIELD #1,8 AS D$
1200 INPUT "D#=";D#
1300 LSET D$=MKD$(D#)
1400 PUT #1,1
1500 GET #1,1
1600 D#=CVD(D$)
1700 PRINT D#
1800 CLOSE #1
1900 END

結果：D#=? 952.32145687
       952.32145687
```

関連命令

CVI／CVL／CVS／CVD関数：文字列を数値データに変換します。
MKSMBF$／MKDMBF$関数：数値をマイクロソフト内部形式の文字列に変換します。
STR$関数　：数値を数字の文字列に変換します。
VAR関数　：数字の文字列を数値に変換します。

MKSMBF$/MKDMBF$関数

エム・ケー・エス・エム・ビー・エフ・ダラー/ *make single Microsoft binary format*/
エム・ケー・ディ・エム・ビー・エフ・ダラー関数 *make double Microsoft binary format*

文字列処理・データファイル

MKSMBF$(単精度型実数)
MKDMBF$(倍精度型実数)

機　能　数値を文字列に変換します。

説　明　内部表現形式の数値(IEEE形式)を、旧形式(マイクロソフト内部形式)に変換し、それを文字列として扱います。

F-BASIC86HG V1.2以前のF-BASICで使用するランダムファイルに、数値データを書き出すときに使用する関数です。

MKSMBF$　:単精度型実数の数値を4バイトの文字列に変換します。
MKDMBF$　:倍精度型実数の数値を8バイトの文字列に変換します。

関連命令　CVSMBF/CVDMBF関数:マイクロソフト内部形式の文字列データを数値データに変換します。

M

MOUSE

マウス *mouse*

1

```
MOUSE 0〔, マウス表示ページ〕
```

機　能　マウス機能を初期化します。

説　明　マウスを使用するときは、必ず最初にMOUSE　0　命令を実行します。

<マウス表示ページ>……マウスカーソルを描画するページを指定します。省略すると、0を設定したとみなされます。

●<マウス表示ページ>はグラフィック2画面モードの時に有効です。

●マウス機能を初期化すると、設定値は次のようになります。

表示位置　：オリジナルスクリーン座標（0, 0）の位置に設定します。

マウスカーソルの形状：左上を指す矢印で、読み取り位置を、カーソルの左上隅の（0, 0）に設定します。

移動比率　：水平方向および垂直方向ともに8を設定します。

移動範囲　：オリジナルスクリーン座標で示される次の範囲に設定します。

画面モード	水平方向	垂直方向
16色モード	0～639	0～479
256色モード	0～639	0～479
32768色モード	0～319	0～239

ボタンが押された回数：0に設定します。

水平、垂直移動量　：0に設定します。

2 MOUSE 1〔,〔sx〕〔,〔sy〕〔, {0 / 1}〕〕〕

機　能　　マウスカーソルの位置の設定や、表示または消去の選択を行います。

説　明　　<sx> <sy>　　……マウスカーソルの表示位置をオリジナルスクリーン座標で指定します。

<sx>：水平位置を16色モードまたは256モードでは0～1023の範囲で、32768色モードでは0～511の範囲で指定します。

<sy>：垂直位置を16色モードまたは256モードでは0～511の範囲で、32768色モードでは0～255の範囲で指定します。

<0><1>　　……カーソル表示の有無を指定します。

0：マウスカーソルを表示しません。

1：<sx><sy>の位置にマウスカーソルを表示します。

省略すると、0を指定したとみなされます。

M

3

MOUSE　2, andパターン, ドットパターン
〔, 水平読み取り位置, 垂直読み取り位置〕

機　能　マウスカーソルの形状と、カーソル内の読み取り位置の設定を行います。

説　明　マウスカーソルの形状は、横16ドット×縦16ドットで構成され、32バイトの文字列<andパターン>と<ドットパターン>を重ね合わせて表示します。
（☞「●マウスカーソルの形状の設定」p.80参照）

<andパターン>……32バイトの文字列を指定します。

<ドットパターン>……32バイトの文字列を指定します。

<水平読み取り位置>……0～15の数値でマウスカーソル内の読み取り位置を指定します。
省略すると、0を指定したとみなされます。

<垂直読み取り位置>……0～15の数値でマウスカーソル内の読み取り位置を指定します。
省略すると、0を指定したとみなされます。

- <andパターン>, <ドットパターン>共に, 32バイトに満たない場合には、残りのバイトには&H00が入り、32バイトを超えた分は無視されます。
- 初期化状態のマウスカーソルの形状を文字列で表すと、以下のようになります。
（☞「●マウスカーソルの形状の設定」p.80参照）

<andパターン>
　CHR$（&H00, &H1F, &H00, &H1F, …&HFF, &HE3, &HFF, &HF7）
<ドットパターン>
　CHR$（&H00, &H00, &H7F, &HC0, …&H00, &H08, &H00, &H00）

関連命令　MOUSE 6：マウスカーソルの形状を単色またはカラー32×32ドットに設定します。

4 MOUSE　3, 移動方向, 移動比率

機　能　マウスカーソルの移動比率を設定します。

説　明　<移動方向>……どの方向の移動比率を設定するかを0または1で指定します。

0：水平方向の<移動比率>を設定します。

1：垂直方向の<移動比率>を設定します。

<移動比率>……マウスカーソルをある一定距離移動させるのに、どれだけマウス本体を移動させるのかを1～255の数値で指定します。
この値が大きいほど、マウスカーソルの動きが鈍くなります。

- MOUSE　0　を実行した直後（初期値）の移動比率は、8です。
- 256以上の移動比率が指定されたときは、255とみなします。

5

```
MOUSE 4, sx1, sy1, sx2, sy2
```

機　能　マウスカーソルの移動範囲を指定します。

説　明　マウスカーソルは、オリジナルスクリーン座標で指定された四角形の範囲内を移動することができます。

＜sx1＞……水平位置の最小値を指定します。

＜sy1＞……垂直位置の最小値を指定します。

＜sx2＞……水平位置の最大値を指定します。

＜sy2＞……垂直位置の最大値を指定します。

●指定できるのは、以下の範囲です。

画面モード	水平方向	垂直方向
16色モード	0～1023	0～511
256色モード	0～1023	0～511
32768色モード	0～511	0～255

●マウスカーソルの移動範囲を表示画面より大きく設定すると、マウスカーソルが表示されない部分ができます。

6 MOUSE 5

機　能　マウスの使用を終了し、マウスカーソルを画面から消します。

説　明　マウスの使用を終了するときは、必ずこの命令を実行します。再びマウスを使用するときは、必ずMOUSE　0　で初期化を行ってください。

例　題　マウスのボタンを押して次々に点を指定し、点と点との間に直線を引きます。Y座標値MOUSE（1）が350を超えると、マウスの動作を終了し、プログラムを終了します。

```
1000  MOUSE 0          ' マウスの初期化を行う.
1100  CLS
1200  MOUSE 1,0,0,1    ' マウスカーソルの位置を決め, 表示する.
1300  IF MOUSE(2,0) THEN 1500
1400  GOTO 1300
1500  X1=MOUSE(0):Y1=MOUSE(1)
1600  IF Y1>350 THEN 2600
1700  IF MOUSE(2,0) THEN 1900
1800  GOTO 1700
1900  X2=MOUSE(0):Y2=MOUSE(1)
2000  IF Y2>350 THEN 2600
2100  MOUSE 1,,,0
2200  CONNECT(X1,Y1)-(X2,Y2)
2300  MOUSE 1,,,1
2400  X1=X2:Y1=Y2
2500  GOTO 1700
2600  MOUSE 5
2700  END
```

関連命令

MOUSE関数　：マウスの設定を行います。
ON　MOUSE（n）GOSUB　：マウス割り込みルーチンを定義します。
MOUSE　ON／OFF／STOP：マウスによる割り込みの許可、禁止または停止を行います。

7 MOUSE 6, モード, andパターン, ドットパターン
〔, 水平読み取り位置, 垂直読み取り位置〕

機　能　マウスカーソルの形状を単色またはカラー32×32ドットに設定します。

説　明　<モード>………………マウスカーソルのモードを設定します。
0 ：単色のマウスカーソルに設定します。
1 ：カラーのマウスカーソルに設定します。

< andパターン> ……DIM命令で定義されている配列名で指定します。
128バイトのサイズが必要です。

<ドットパターン>……DIM命令で定義されている配列名で指定します。
単色マウスカーソルの場合は128バイト、カラーマウスカーソルで16色モードの時は512バイト、256色モードの時は1024バイト、32768色モードの時は2048バイトのサイズが必要です。
GET@Aで32ドットのパターンを取り込んだ配列がそのまま使用できます。

<水平読み取り位置>… 0～31の数値でマウスカーソル内の読み取り位置を指定します。
省略すると、0を指定したとみなされます。

<垂直読み取り位置>… 0～31の数値でマウスカーソル内の読み取り位置を指定します。
省略すると、0を指定したとみなされます。

● < andパターン >、<ドットパターン>共に、配列がパターンより小さい場合は、残りのバイトに&H00が入り、超えた部分は無視されます。

MOUSE関数

マウス関数 *mouse*　　マウス

1　MOUSE （式）

機　能　マウスカーソルの情報を返します。

説　明　マウスカーソルの現在のオリジナルスクリーン座標上の位置、移動量の情報を返します。

<式>…… 0、1、9または10のいずれかを指定します。

0：マウスカーソルの現在の水平位置をオリジナルスクリーン座標で返します。

1：マウスカーソルの現在の垂直位置をオリジナルスクリーン座標で返します。

9：この関数を呼び出してから，今回呼び出すまでのマウスカーソルの水平移動量を返します。ただし、MOUSE 1で表示位置を指定した場合は、その位置からの水平移動量になります。
正の値は右方向に、負の値は左方向に移動したことを表します。

10：この関数を呼び出してから、今回呼び出すまでのマウスカーソルの垂直移動量を返します。ただし、MOUSE 1で表示位置を指定した場合は、その位置からの垂直移動量になります。
正の値は下方向に、負の値は上方向に移動したことを表します。

コンパイラ　●コンパイラでは、<式>には定数値のみが有効です。数値式を記述することはできません。

M

2 MOUSE（式, ボタン番号）

機　能　マウスのボタンの状態、またはマウスカーソルの情報を返します。

説　明　＜ボタン番号＞で指定されたマウスボタンについて、＜式＞で指定された情報を返します。

＜式＞　……2～8のいずれかを指定します。

2：＜ボタン番号＞で指定したボタンが押されていれば－1を、放されていれば0を返します。

3：最後にこの関数を呼び出してから、今回呼び出すまでに＜ボタン番号＞で指定したボタンが押された回数を返します。

4：＜ボタン番号＞で指定したボタンが最後に押されたときの、マウスカーソルの水平位置を返し、＜ボタン番号＞で指定したボタンが押された回数を初期化します。

5：＜ボタン番号＞で指定したボタンが最後に押されたときの、マウスカーソルの垂直位置を返し、＜ボタン番号＞で指定したボタンが押された回数を初期化します。

6：最後にこの関数を呼び出してから、今回呼び出すまでに＜ボタン番号＞が放された回数を返します。

7：＜ボタン番号＞で指定したボタンが最後に放されたときの、マウスカーソルの水平位置を返し、＜ボタン番号＞で指定したボタンが押された回数を初期化します。

8：＜ボタン番号＞で指定したボタンが放されたときの、マウスカーソルの垂直位置を返し、＜ボタン番号＞で指定したボタンが押された回数を初期化します。

＜ボタン番号＞…… 0 または 1 でボタンを指定します。

0 ：マウスの左ボタンを示します。

1 ：マウスの右ボタンを示します。

コンパイラ　●コンパイラでは、＜式＞には定数値のみが有効です。数値式を記述することはできません。

例　題　マウスの左ボタンを押して次々に点を指定し、点と点の間に直線を引きます。Y座標値MOUSE（1）が350を超えると、マウスの動作を終了し、プログラムを終了します。

```
1000  MOUSE 0          'マウスの初期化を行う.
1100  CLS
1200  MOUSE 1,0,0,1
1300  IF MOUSE(2,0) THEN 1500
1400  GOTO 1300
1500  X1=MOUSE(0):Y1=MOUSE(1)
1600  IF Y1>350 THEN 2600
1700  IF MOUSE(2,0) THEN 1900
1800  GOTO 1700
1900  X2=MOUSE(0):Y2=MOUSE(1)
2000  IF Y2>350 THEN 2600
2100  MOUSE 1,,,0
2200  CONNECT(X1,Y1)-(X2,Y2)
2300  MOUSE 1,,,1
2400  X1=X2:Y1=Y2
2500  GOTO 1700
2600  MOUSE 5
2700  END
```

関連命令

MOUSE ：マウス機能の設定を行います。

ON　MOUSE（n）GOSUB ：マウス割り込みルーチンを定義します。

MOUSE　ON／OFF／STOP ：マウスによる割り込みの許可、禁止または停止を行います。

M

MOUSE(n)ON/OFF/STOP

マウス・オン/オフ/ストップ　*mouse(n) on/ off/ stop*

プログラムの分岐・マウス

```
                     ON
MOUSE（条件番号）    OFF
                     STOP
```

機　能　マウスによる割り込みの許可、禁止および停止を行います。

説　明　＜条件番号＞……マウスによる割り込みが発生する条件を1～5の数値で指定します。

1：マウスを動かす。
2：左ボタンを押す。
3：左ボタンを放す。
4：右ボタンを押す。
5：右ボタンを放す。

● MOUSE（条件番号）ON
命令実行後は、＜条件番号＞で指定した状態になると割り込みが発生し、ON MOUSE（n）GOSUB命令で定義されている割り込み処理ルーチンに、制御が移されます。

● MOUSE（条件番号）OFF
命令実行後は、＜条件番号＞で指定した状態での割り込みが禁止されます。＜条件番号＞で指定した状態になっても割り込みは発生しません。

● MOUSE（条件番号）STOP
命令実行後は、＜条件番号＞で指定した状態での割り込みを一時停止します。＜条件番号＞で指定した状態になっても割り込み処理ルーチンへ制御は移されませんが、その後同じ＜条件番号＞のMOUSE（条件番号）ON命令で割り込みが許可されれば、割り込み処理ルーチンが実行されます。

● MOUSE（条件番号）ON、OFFまたはSTOP命令を実行する前に、ON MOUSE（n）GOSUB命令を実行し、割り込み処理ルーチンの定義をしておく必要があります。

例　題　左ボタンを押し、マウス割り込みルーチンの実行を行います。

```
1000  CLS  : A=1
1100  MOUSE 0
1200  ON MOUSE(2) GOSUB 1800    ' 左ボタンを押すと，1800行に移る．
1300  MOUSE(2) ON               ' 左ボタンを押したとき，割り込みを許可
1400  IF A>7 THEN *END ELSE COLOR A
1500  PRINT "*" ; :GOTO 1500
1600  *END
1700  MOUSE(2) OFF :END        ' 左ボタンを押したときの割り込みを禁止
1800  '
1900  A=A+1
2000  RETURN 1400
```

関連命令　ON　MOUSE（n）GOSUB：割り込み処理ルーチンを定義します。

M

MOVIE CLOSE

ムービー・クローズ　*movie close*　　動画／アニメーション

```
MOVIE CLOSE
```

機　能　OPENしたムービーファイルをCLOSEします。

説　明　ムービーファイルに付随するすべてのメモリを開放します。ただし、DLL領域は開放されません。DLL領域が開放されるのは、CLEAR命令を実行した時、またはプログラムを終了したときです。

●V1.1のBASICでは、この命令は使用できません。

M

MOVIE INFO

ムービー・インフォメーション *movie information*　　動画／アニメーション

1　MOVIE INFO 1, ロング型整数配列名

機　能　OPENしたムービーファイルの情報を得ます。

説　明　＜配列の数字＞

	動画ファイル(.MVE)	アニメーションファイル(.MMM)
1	フレーム／秒	無効(０)
2	フレーム数	フレーム数
3	総再生時間(秒)	総再生時間(秒)
4	トラック数	無効(０)
5	再生開始タイミング	無効(０)
6	画枠　X(max)	画枠　X(max)
7	Y(max)	Y(max)

●V1.1のBASICでは、この命令は使用できません。

M

2 MOVIE　INFO　2，文字列配列名

機　能　OPENしたムービーファイルの情報を得ます。

説　明　＜配列の数字＞

	動画ファイル(.MVE)	アニメーションファイル(.MMM)
1	タイトル	タイトル
2	作成日	無効(0)
3	作成者	無効(0)
4	最終更新日	無効(0)
5	最終更新者	無効(0)
6	著作権表示	無効(0)

●V1.1のBASICでは、この命令は使用できません。

MOVIE OPEN

ムービー・オープン　*movie open*　　　　動画／アニメーション

```
MOVIE OPEN "ファイル名.拡張子"
```

機　能　　ムービーファイルをOPENします。

説　明　　<ファイル名>……ムービーファイルのファイル名を指定します。

<拡張子>…………ムービーファイルの属性を指定します。

mve　：動画ファイル

mmm：アニメーションファイル

●V1.1のBASICでは、この命令は使用できません。

MOVIE PLAY

ムービー・プレイ　*movie play*　　動画／アニメーション

```
MOVIE　PLAY〔, 開始フレーム〔, 終了フレーム〕〕
```

機　能　ムービーファイルを再生します。

説　明　＜開始フレーム＞……どのフレームから再生するかを指定します。
省略すると、ファイルの先頭から再生されます。

＜終了フレーム＞……どのフレームまで再生するかを指定します。
省略すると、ファイルの最後まで再生されます。

- V1.1のBASICでは、この命令は使用できません。
- 動画ファイル(.MVE)は、32768色モードのときのみ再生できます。
- アニメーションファイル(.MMM)は、256色モードのときのみ再生できます。

NAME

ネーム　*name*　　データファイル

NAME　旧ファイルディスクリプタ　AS　新ファイル名

機　能　ファイル名を変更します。

説　明　<旧ファイルディスクリプタ>のファイルを<新ファイル名>に変更します。

<旧ファイルディスクリプタ>……現在のファイル名を指定します。
ファイルディスクリプタの<オプション>は指定できません。

<新ファイル名>……新しいファイル名を指定します。
パス名は指定できません。

- <旧ファイルディスクリプタ>のファイルは、クローズしていなければなりません。
- BASICエディタのファイルの「名前変更」を指定した場合と同じ結果になります。

例　題　ファイル名“DATA1DAT”を“DATA2DAT”に変更します。

```
NAME "DATA1DAT" AS "DATA2DAT"
```

NEW

ニュー *new*　　コマンド

```
NEW
```

機　能　メモリ上のプログラムを消去し、全ての変数を初期化します。

説　明
- NEW命令を実行すると、オープンされていたファイルは全てクローズされます。
- NEW命令の実行後は、BASICエディタに戻ります。
- BASICエディタの「その他」の「新規作成」を指定した場合と同じ結果になります。

コンパイラ
- コンパイラでは、NEW命令は使用できません。NEW命令を含むプログラムをコンパイルするとエラーになります。

NEXT

ネクスト *next*

プログラムの分岐

```
NEXT 〔制御変数〔, 制御変数〕…〕
```

機　能　FOR～NEXTループの終わりを示します。

説　明　<制御変数>……対応するFOR命令で指定された制御変数を指定します。省略すると、他のNEXT命令と対応付けられていない直前のFOR命令と対応付けられます。

- 1つのNEXT命令を、複数のFOR命令と対応付け、一度に複数個のFOR～NEXTループを終了させることができます。
  このとき、<制御変数>は、内側のループのFOR命令に対応する制御変数から外側のループの制御変数へと、順番に指定していきます。

例　題　FOR命令の例題を参照してください。

関連命令　FOR　：NEXT命令とループを作り、制御変数の値を変えながら一連の命令を繰り返し実行します。

N

OCT$

オクタル・ダラー　*octal$*　　計算

```
OCT$(式)
```

機　能　整数を8進数の文字列に変換します。

説　明　<式>……−2147483648〜+4294967295の範囲の整数値を設定します。

- 変換結果は"0"〜"37777777777"になります。
- <式>が小数部分を含む場合は、内部でINT関数を実行し、整数化してから変換します。
- 負の値を指定すると、内部表現の補数が8進数に変換され、正の値になります。

例　題　入力した値を8進数に変換し表示します。

```
1000 INPUT A
1100 B$=OCT$(A)
1200 PRINT B$
1300 END

結果：? 12
       14
```

関連命令　HEX$：整数を16進数の文字列に変換します。

ON COM(n) GOSUB

オン・コム・ゴー サブ　*on communication(n) go to subroutine*　プログラムの分岐

ON　COM（ポート番号）GOSUB ｛行番号／ラベル名｝

機　能　通信回線からの入力割り込み時の割り込み処理ルーチンを定義します。

説　明　<ポート番号>……通信ポート番号を0～4の数値で指定します。

0：本体内のRS-232Cポート
1：増設RS-232Cインタフェースの通信ポート1
2：増設RS-232Cインタフェースの通信ポート2
3：増設RS-232Cインタフェースの通信ポート3
4：増設RS-232Cインタフェースの通信ポート4

<行番号> <ラベル名> ……入力割り込みの発生時に呼び出される割り込み処理ルーチンの開始行を、指定します。

- 割り込み処理ルーチンの実行中は、自動的にCOM（n）STOPに設定されます。したがって、この間に同じ通信ポートからの入力があっても、割り込み動作は行われず、保留されます。ルーチンの処理が終了した時点で、自動的にCOM（n）ONに設定されます。（ただし、割り込み処理ルーチン内にCOM（n）OFFがある場合は、ルーチンの処理後、COM（n）ONには設定されません。）
- エラー処理ルーチンの実行中は、全ての割り込みが停止状態になるため、通信ポートからの入力があっても割り込み動作は行われず保留されます。
- 割り込み処理ルーチンからの復帰は、RETURN命令で行います。RETURN命令に行指定がある場合は、指定した行からプログラムが再開されます。行指定がない場合は、中断した箇所から再開されます。
- CHAIN命令を実行すると、割り込み処理ルーチンの定義は無効となりますので、連結されたプログラムの中で定義しなおす必要があります。

O

例　題　本体内のRS−232CポートCOM0に入力があった場合、行番号2220から始まる割り込み処理ルーチンを実行します。

```
1000 OPEN "O",#1,"COM0:"
1100 OPEN "I",#2,"COM0:"
1200 ON COM(0) GOSUB 2200
1300 COM(0) ON
1400 CLS
1500 LOCATE 0,0,1
1600 A$="":WHILE A$="":A$=INKEY$:WEND
1700 IF A$=CHR$(&H0D) THEN A$=A$+CHR$(&H0A)
1800 PRINT #1,A$;
1900 GOTO 1600
2000 CLOSE #1,#2
2100 END
2200 WHILE NOT EOF(2)
2300 B$=INPUT$(1,2)
2400 IF B$=CHR$(&H0D) THEN B$=B$+CHR$(&H0A)
2500 PRINT B$;
2550 WEND
2600 RETURN
```

関連命令　COM (n) ON／OFF／STOP：通信回線からの割り込みを許可、禁止または停止します。

ON ERROR GOTO

オン・エラー・ゴーツー　*on　error　goto*　　　　プログラムの分岐・エラーの処理

```
                ┌行番号
ON  ERROR  GOTO ┤
                └ラベル名
```

機　能　エラー処理ルーチンを定義し、エラートラップを許可します。

説　明　エラーが検出されたとき、ON　ERROR　GOTO命令で指定したエラー処理ルーチンが呼び出されます。

ON　ERROR　GOTO命令が実行されていない場合にエラーが発生すると、エラーメッセージが表示され、プログラムは実行を停止します。

＜行番号＞＜ラベル名＞……エラー処理ルーチンの実行開始行を指定します。

- エラーが発生したときには、エラー番号とエラーの発生した行の行番号が、ERRとERLに自動的に代入されます。
- エラー処理ルーチンの中ではエラートラップは発生しません。もし、エラー処理ルーチンの中でエラーが発生した場合は、エラーメッセージを表示し、プログラムの実行を停止します。
- エラートラップを禁止する場合は、ON　ERROR　GOTO　0を指定します。
- エラー処理ルーチンからの復帰は、RESUME命令で行います。
- CHAIN命令を実行すると、エラー処理ルーチンの定義は無効となりますので、連結されたプログラムの中で定義しなおす必要があります。

O

例　題　入力した数値が3桁以上のときに、エラー処理ルーチンを実行します。エラー処理ルーチンでは、エラーを起こした行番号を画面に表示します。

```
1000 ON ERROR GOTO 1700
1100 RANDOMIZE (TIME/4)
1200 A=INT(RND(1)*9)+1
1300 INPUT B
1400 IF A<B THEN ERROR 80 ELSE IF A>B THEN ERROR 81
1500 PRINT "そのとおり！！"
1600 END
1700 IF ERR=80 THEN PRINT "大きすぎ！" ELSE PRINT "小さすぎ！"
1800 RESUME 1300
1900 END
```

```
1000 ON ERROR GOTO *ERREXIT
1100 PRINT "2桁までの数字を入力してください "
1200 *INPUTP
1300   INPUT A
1400   IF A>99 THEN ERROR 99
1500   PRINT "O K !"
1600   END
1700 *ERREXIT
1800   IF ERR=99 THEN PRINT USING"( 行番号#### ) ";ERL
1900 RESUME *INPUTP
2000   END
```

```
結果 2桁までの数字を入力してください
     ? 100
     ( 行番号1400 )
     ? 10
     O K !
```

関連命令

ERROR　：エラー発生のシミュレートや、使用者によるエラー番号の定義を行います。

ERR／ERL：発生したエラーのエラー番号および行番号を返します。

RESUME　：エラー処理ルーチンから戻り、プログラム実行を再開します。

ON～GOSUB

オン・ゴーサブ *on～go to subroutine*

プログラムの分岐

```
ON 式 GOSUB {行番号|ラベル名} [, {行番号|ラベル名}] …
```

機　能　式の値により、対応する行のサブルーチンを呼び出します。

説　明　<式>の値が1のとき1番目の<行番号>または<ラベル名>に、2ならば2番目の<行番号>または<ラベル名>に、というように、式の値により分岐します。

<式> ……結果が正の値になる数値式を指定します。

<行番号> <ラベル名> ……呼び出したいサブルーチンの実行開始行を指定します。

- <式>の値が整数でないときは、小数部を四捨五入した整数に変換して処理されます。
- <式>の値が0のとき、および<行番号>または<ラベル名>の個数より大きいときは、次の行に制御が移ります。
- <式>の値が負のときはエラーになります。
- サブルーチンからの復帰は、RETURN命令で行います。RETURN命令に行指定がある場合は、指定した行からプログラムを実行します。行指定がない場合は、ON～GOSUB文の次の行を実行します。

例　題　入力する値により、呼び出すサブルーチンを分岐させ実行します。4を入力すると処理が終了します。

```
1000 INPUT A
1100 ON A GOSUB 1300, 1400, 1500, 1600  ' 1が入力されたら1300行へ, ・・・
1200 GOTO 1000
1300 PRINT "富士通":RETURN
1400 PRINT "F-BASIC":RETURN
1500 PRINT "386":RETURN
1600 END

結果：? 1
      富士通
      ? 2
      F-BASIC
      ? 3
      386
      ? 4
```

関連命令　ON～GOTO：式の値により、指定された行番号の1つに分岐します。

ON～GOTO

オン・ゴーツー　*on～goto*　　　　プログラムの分岐

```
ON 式 GOTO {行番号  } [ , {行番号  } ] …
           {ラベル名}       {ラベル名}
```

機　能　式の値により、指定された行番号の１つに分岐します。

説　明　＜式＞の値が１のとき１番目の＜行番号＞または＜ラベル名＞に、２ならば２番目の＜行番号＞または＜ラベル名＞に、というように、式の値により分岐します。

＜式＞……結果が正の値になる数値式を指定します。

＜行番号＞＜ラベル名＞……分岐したい行を指定します。

- ＜式＞の値が整数でないときは、小数部を四捨五入した整数に変換して処理されます。
- ＜式＞の値が０のとき、または＜行番号＞または＜ラベル名＞の個数より大きいときは、次の行に制御が移ります。
- ＜式＞の値が負のときはエラーになります。

例　題　入力する値により＊SUB1、＊SUB2または＊ENDのいずれかに分岐します。3を入力すると、処理を終了します。

```
1000 *MAIN
1100  INPUT B
1200  ON B GOTO *SUB1,*SUB2, *END:GOTO *MAIN
1300 *SUB1
1400  PRINT "SUB-1":GOTO *MAIN
1500 *SUB2
1600  PRINT "SUB-2":GOTO *MAIN
1700 *END
1800  PRINT "END":END

結果：? 1
      SUB-1
      ? 2
      SUB-2
      ? 3
      END
```

関連命令　ON～GOSUB：式の値により、指定された行のサブルーチンを呼び出します。

ON INTERVAL GOSUB

オン・インターバル・ゴーサブ　*on interval go to subroutine*　プログラムの分岐・時計

```
                          ┌ 行番号 ┐
ON  INTERVAL  GOSUB ┤        ├
                          └ ラベル名 ┘
```

機　能　INTERVAL命令で指定した割り込み間隔で実行されるインターバルタイマ割り込み処理ルーチンを定義します。

説　明　<行番号> <ラベル名> ……インターバルタイマ割り込み処理ルーチンの開始行を指定します。

- 割り込み間隔は、ON　INTERVAL　GOSUB命令の実行前に、INTERVAL命令で指定します。
- インターバルタイマ割り込み処理ルーチンの実行中は、自動的にINTERVAL　STOP状態に設定されます。したがって、この間にインターバルタイマによる割り込みがかかっても、割り込み動作は行われず保留されます。
  ルーチンの処理が終了した時点で、自動的にINTERVAL　ON状態が設定されます。(ただし、インターバルタイマ割り込み処理ルーチン内にINTERVAL　OFF命令がある場合は、ルーチンの処理後、自動的にINTERVAL　ON状態には設定されません。)
- エラー処理ルーチンの実行中は、その他の全ての割り込みが停止状態に設定されているため、インターバルタイマによる割り込みがかかっても割り込み動作は行われず保留されます。
- インターバルタイマ割り込み処理ルーチンからの復帰はRETURN命令で行います。RETURN命令に行指定がある場合は、指定した行からプログラムが再開されます。行指定がない場合は、中断した箇所から再開されます。
- CHAIN命令を実行すると、割り込み処理ルーチンの定義は無効となりますので、連結されたプログラムの中で定義しなおす必要があります。

O

例　題　箱の色を１秒ごとに塗り替え、７色の表示が終わるとプログラムを終了します。

```
1000  CLS
1100  I=1 :PALETTE 7,7
1200  LINE (100,100)-(200,150),PSET,%7,BF  ' 四角形を書く.
1300  INTERVAL 1                          ' 割り込み間隔を指定する.
1400  ON INTERVAL GOSUB 2200              ' 割り込みルーチンを定義する
1500  INTERVAL ON                         ' 割り込みを許可する.
1600  GO TO 1600
1700  '
1800  *END
1900  INTERVAL OFF                        ' 割り込みを停止する.
2000  END
2100  '
2200  I=I+1
2300  IF I<=7 THEN 2400 ELSE RETURN *END
2400  PALETTE 7,I
2500  RETURN 1600
```

関連命令　INTERVAL　　　：インターバルタイマ割り込み間隔を指定します。

INTERVAL ON／OFF／STOP：インターバルタイマ割り込みを許可、禁止または停止します。

ON KEY(n)GOSUB

オン・キー・ゴーサブ　*on key(n) go to subroutine*　　プログラムの分岐・PFキー

ON　KEY　（PFキー番号）　GOSUB ｛行番号／ラベル名｝

機　能　PFnキーを押すと実行されるPFキー割り込み処理ルーチンを定義します。

説　明　＜PFキー番号＞……押すと、割り込みを発生させるPFキーを1～10の数値で指定します。

＜行番号＞＜ラベル名＞……割り込み処理ルーチンの実行開始行を指定します。

- キー割り込み処理ルーチンの実行中は、自動的にKEY（n）STOPに設定されています。したがって、実行中に同じPF（n）キーによる割り込みがかかっても、割り込み動作は行われず保留されます。割り込み処理ルーチンの実行が終了した時点で、自動的にKEY（n）ONが設定されます。（ただし、割り込み処理ルーチン内にKEY（n）OFF命令がある場合は、割り込み処理ルーチンの実行終了後、自動的にKEY（n）ONには設定されません。）
  別のPF（n）キーによる割り込みは、実行可能です。
- エラー処理ルーチンの実行中は，その他の全ての割り込みが停止状態になるため、キー割り込みがかかっても割り込み動作は行われず保留されます。
- 割り込み処理ルーチンからの復帰は、RETURN命令で行います。
  RETURN命令に行指定がある場合は、指定した行から再開されます。行指定がない場合は、中断した箇所から再開されます。
- CHAIN命令を実行すると、割り込み処理ルーチンの定義は無効となりますので、連結されたプログラムの中で、定義しなおす必要があります。

O

例　題　PF1キーを押すことによって、画面に表示する“＊”の色を変更します。
PF1キーを7回押すと、プログラムを終了します。

```
1000  CLS
1100  A=1
1200  ON KEY (1) GOSUB 2200          ' PF1 を押すと，2200行に処理が移る．
1300  KEY (1) ON                     ' PF1 からの割り込みを許可する．
1400  IF A>7 THEN *END  ELSE COLOR A
1500  WHILE -1
1600    PRINT "*";
1700  WEND
1800  *END
1900  KEY (1) OFF                    ' PF1 からの割り込みを禁止する．
2000  IF INKEY$<>"" THEN 2000 ELSE END
2100  '
2200  A=A+1
2300  RETURN 1400
```

関連命令　KEY　ON／OFF／STOP：PFキーからの割り込みを許可、禁止または停止します。

ON MOUSE(n) GOSUB

オン・マウス・ゴーサブ　*on mouse(n) go to subroutine*　　プログラムの分岐・マウス

ON　MOUSE　（条件番号）　GOSUB　{行番号 / ラベル名}

機　能　<条件番号>の割り込みが発生したときに呼び出す割り込み処理ルーチンを定義します。

説　明　<条件番号>　……1～5の数値で、割り込み発生原因を指定します。

1：マウスを動かす。
2：左ボタンを押す。
3：左ボタンを放す。
4：右ボタンを押す。
5：右ボタンを放す。

<行番号><ラベル名>……割り込み処理ルーチンの実行開始行を指定します。

- マウス割り込み処理ルーチンの実行中は、自動的に割り込み停止（STOP）状態に設定されます。同じ条件番号による割り込みがあっても、割り込み動作は行われず保留されます。割り込み処理ルーチンの処理が終了した時点で、自動的に割り込み許可（ON）状態に設定されます。別の条件番号による割り込みは可能です。（ただし、マウス割り込み処理ルーチン内にMOUSE（n）OFF命令がある場合は、ルーチンの処理後、自動的に割り込み許可（ON）状態には設定されません。）
- エラー処理ルーチンの実行中は、全ての割り込みが停止状態になるため、マウス割り込みがかかっても割り込み動作は行われず保留されます。
- マウス割り込み処理ルーチンからの復帰は、RETURN命令で行います。RETURN命令に行指定がある場合は、指定した行から再開されます。ない場合は、中断した箇所から再開されます。
- CHAIN命令を実行すると、割り込み処理ルーチンの定義は無効となりますので、連結されたプログラムの中で定義しなおす必要があります。

例　題　マウスの左ボタンを押し、画面に表示されている“＊”の色を変えます。左ボタンを７回押すと、プログラムを終了します。

```
1000  CLS  : A=1
1100  MOUSE 0
1200  ON MOUSE(2) GOSUB 1800     ' 左ボタンを押すと，1800行に移る.
1300  MOUSE(2) ON                ' 左ボタンを押したとき，割り込みを許可
1400  IF A>7 THEN *END ELSE COLOR A
1500  PRINT "*" ; :GOTO 1500
1600  *END
1700  MOUSE(2) OFF :END          ' 左ボタンを押したときの割り込みを禁止
1800  '
1900  A=A+1
2000  RETURN 1400
```

関連命令　MOUSE　ON／OFF／STOP：マウスの割り込みの許可、禁止および停止を行います。

ON TIME GOSUB

オン・タイム・ゴーサブ　*on time go to subroutine*　　プログラムの分岐・時計

```
ON  TIME  GOSUB { 行番号
                  ラベル名 }
```

機　能　タイマ割り込み処理ルーチンを定義します。

説　明　TIME命令で指定した割り込み時刻に実行されるタイマ割り込み処理ルーチンを定義します。

<行番号> <ラベル名>……タイマ割り込み処理ルーチンの開始行を指定します。

- 割り込み時刻は、ON　TIME　GOSUB命令の実行前に、TIME命令で指定します。
- タイマ割り込み処理ルーチンの実行中は、自動的に割り込み停止(STOP)状態に設定されます。したがって、この間に次のTIME命令による割り込みがかかっても、割り込み動作は行われず、保留されます。ルーチンの処理が終了した時点で、自動的に割り込み許可（ON）状態が設定されます。（ただし、タイマ割り込みルーチン内にTIME　OFF命令がある場合は、ルーチン終了後、自動的にON状態にはなりません。
- エラー処理ルーチンの実行中は、その他の全ての割り込みが停止状態に設定されているため、タイマ割り込みがかかっても割り込み動作は行われず保留されます。
- タイマ割り込みルーチンからの復帰はRETURN命令で行います。RETURN命令に行指定がある場合は、指定した行からプログラムが再開されます。
  行指定がない場合は、中断した箇所から再開されます。
- CHAIN命令を実行すると、割り込み処理ルーチンの定義は無効となりますので、連結されたプログラムの中で定義しなおす必要があります。

O

例　題　指定した時刻になるまで、時刻の表示を続けます。指定された時刻になるとブザーを鳴らし、時刻を赤色で表示します。

```
1000  CLS
1100  LOCATE 0,1 :PRINT "いまは，";TIME$;"です．"
1200  LINE INPUT "何時に設定しますか．(hh:mm:ss)?";SET$
1300  TIME SET$
1400  ON TIME GOSUB 2200                    '割り込みルーチンを定義する．
1500  TIME ON                               '割り込みを許可する
1600  WHILE -1
1700    LOCATE 0,5 :PRINT"*";TIME$;"*"
1800  WEND
1900  TIME OFF                              '割り込みを停止する
2000  END
2100  '
2200  BEEP 1
2300  COLOR 2 :LOCATE 0,5 :PRINT "*";TIME$;"*"
2400  FOR T=1 TO 2000 :NEXT T
2500  BEEP 0 :COLOR 7
2600  RETURN 1900
```

```
結果：いまは，17:24:06です．

      何時に設定しますか．(hh:mm:ss)? 17:25:00

      *17:25:00*
```

関連命令　TIME　　　　　　　　　：タイマ割り込みの時刻を設定します。
　　　　　TIME　ON／OFF／STOP：タイマ割り込みを許可、禁止または停止します。

OPEN

オープン　*open*　　データ入力・データファイル

1　OPEN　モード，〔#〕ファイル番号，ファイルディスクリプタ

機　能　ファイルのオープン処理を行います。

説　明　<ファイルディスクリプタ>で指定したファイルをオープンし、<ファイル番号>とバッファを割り当てます。

<モード>　……ファイルの用途を指定します。

モード	説　　明
"I"	（シーケンシャルファイルの入力モード） シーケンシャルファイルからの入力処理を行うモードです。指定したファイルは、既に存在していなければなりません。
"O"	（シーケンシャルファイルの出力モード） シーケンシャルファイルを新たに作成し、出力処理を行うモードです。指定したファイルは、存在していてはいけません。
"A"	（シーケンシャルファイルの追加モード） シーケンシャルファイルにデータの追加を行うモードです。指定したファイルが既に存在していればデータの最後に新しいデータを追加し、存在していなければ、ファイルを新しく作成し、データの出力を行います。
"R"	（ランダム入出力モード） ランダムファイルの入出力処理を行うモードです。この指定は、ディスク上に存在するファイルに対してのみ有効で、存在していないときは、指定したファイル名で新しいファイルを作成します。

O

<#> ……省略しても意味は変わりません。

<ファイル番号>……オープンするファイルに割り当てる番号で、1～16までの数値で指定します。

（☞「●ファイル番号」 p.35参照）

<ファイルディスクリプタ>……オープンするファイルを指定します。

（☞「●ファイルディスクリプタの形式」 p.31参照）

●<ファイル番号>でファイルを指定してデータ入出力などを行う場合には、事前にそのファイルをオープンして、ファイル番号を定義し、バッファを用意する必要があります。

> **注意** オープンしたファイルは、プログラム終了までに必ずクローズ処理を行う必要があります。
> 特にBREAKキーなどでプログラムの実行を中断した場合には忘れずに、オープンしたファイルをクローズしてください。

例　題　キーボードからデータを入力し、プリンタに出力します。

```
1000 OPEN "I",#1,"KYBD:"        ' キーボードを入力モードでオープンする
1100 OPEN "O",#2,"LPT0:"        ' プリンタを出力モードでオープンする
1200 INPUT#1,D$
1300 IF D$="END" THEN 1600      ' END が入力されたら終了する
1400 PRINT#2,D$
1500 GOTO 1200
1600 CLOSE#1,#2
1700 END
                                    プリンタへの出力
結果：? FUJITSU          →          FUJITSU
      ? 386AV                        386AV
      ? SERIES                       SERIES
      ? END
```

2

```
OPEN ファイルディスクリプタ 〔OPENモード〔ALLOWモード〕〕
                         AS 〔#〕ファイル番号
```

機　能　ファイルのオープン処理を行います。

説　明　OPEN命令形式1と、同様の処理を行います。
形式1と異なる点は、モードの指定方法です。その他の項目は、形式1と同様に指定します。

<OPENモード>……ファイルの用途に従って以下の中から指定します。省略した場合は、ランダムファイルの入出力処理を指定したとみなされます。

モード	説　　明
FOR INPUT	（入力モード） シーケンシャルファイルからの入力処理を行うモードです。指定したファイルは、既に存在していなければなりません。
FOR OUTPUT	（出力モード） シーケンシャルファイルを新たに作成し、出力処理を行うモードです。指定したファイルは、存在していてはいけません。
FOR APPEND	（追加モード） シーケンシャルファイルにデータの追加を行うモードです。指定したファイルが既に存在していればデータの最後に新しいデータを追加し、存在していなければ、ファイルを新しく作成し、データの出力を行います。
FOR RNDIO	（ランダム入出力モード） ランダムファイルに対する入出力処理を行う。 指定したファイルが存在しない場合は、新規ファイルを作成する。

＜ALLOWモード＞……同一ファイルに他のプログラムから行う、後続のオープン処理で、どのモードを認めるかを指定します。

モード	説　　明
ALLOW INPUT	（入力許可） 後続のオープン処理で、入力処理（FOR INPUT）だけが可能です。
ALLOW OUTPUT	（出力許可） 後続のオープン処理で、出力処理（FOR OUTPUT）だけが可能です。
ALLOW ALL	（入出力許可） 後続のオープン処理で、入力処理、出力処理の両方が可能です。
ALLOW NONE	（入出力禁止） 後続のオープン処理をすべて禁止します。

● ＜OPENモード＞の指定によっては、指定できない＜ALLOWモード＞があります。＜ALLOWモード＞を省略した場合は、◎を指定したとみなされます。

ALLOW MODE / OPEN MODE	ALLOW INPUT	ALLOW OUTPUT	ALLOW ALL	ALLOW NONE
FOR INPUT	○	×	◎	○
FOR OUTPUT	◎	×	×	○
FOR APPEND	◎	×	×	○
FOR RNDIO	○	○	◎	○
省　略	×	×	×	×

○：指定できます。
×：指定できません。

● V1.1のBASICでは、＜ALLOWモード＞を指定できません。

例　題　キーボードからデータを入力し、プリンタに出力します。

```
1000 OPEN "KYBD:" FOR INPUT AS #1  ' キーボードを入力モードでオープンする
1100 OPEN "LPT0:" FOR OUTPUT AS #2  ' プリンタを出力モードでオープンする
1200 INPUT#1,D$
1300 IF D$="END" THEN * 入力終わり  ' ＥＮＤが入力されたら終了する
1400 PRINT#2,D$
1500 GOTO 1200
1550 *入力終わり
1600 CLOSE#1,#2
1700 END
                                  プリンタへの出力
結果： ? FUJITSU        →          FUJITSU
       ? 386AV                     386AV
       ? TOWNS                     TOWNS
       ? END
```

関連命令　CLOSE：ファイルのクローズ処理を行います。

O

OUT

アウト関数　*out*　　機械語プログラム

```
OUT I／Oアドレス，データ〔，型番号〕
```

機　能　ハードウェアのI／Oレジスタにデータを書き込みます。

説　明　<I／Oアドレス>……データを書き込む位置を、アドレスで指定します。指定できる値は&H0000～&HFFFFです。

<データ>……書き込むデータを以下の範囲で記述します。

型番号　1の場合:0～255
2の場合:−32768%～32767%
4の場合:−2147483648&～2147483647&

<型番号>……書き込みの単位を1,2または4で指定します。それぞれの数値は、以下の意味を持っています。
1:BYTE(1 byte)単位で書き込みを行います。
2:WORD(2 byte)単位で書き込みを行います。
4:DWORD(4 byte)単位で書き込みを行います。
省略すると1を指定したとみなされます。

- 指定した型番号と異なる型のデータを書き込もうとすると、プログラムは正しく動作しません。
- V1.1 L20以前のBASICでは、OUT命令は使用できません。

OUTM

アウト・エム　*output MIDI*　　音楽/音声

```
OUTM 〔#ポート番号,〕 式〔, 式〕…
```

機　能　MIDIポートへデータを送ります。

説　明　<#ポート番号> ……送り出し先のポート番号を指定します。
0〜7：外部MIDIポート
255　：内蔵音源
省略すると、0を指定したとみなされます。

<式>　……送り出すデータを指定します。

- MIDI楽器へ送るデータでは、音の高さと音量を、次のように設定します。
  音の高さ：0〜127の数値で指定します。
  オクターブ4のCの高音が60になります。1大きくなるごとに半音ずつ上がっていきます。
  音量　：0〜127の数値で指定します。
  8×nが、PLAY命令のVn（n=0, 1……, 15）に対応しています。

- MIDI規格に合わないデータを送った場合、エラーメッセージは表示されませんが、MIDI楽器の動作は保証できません。詳しくは、MIDI規格のデータフォーマット（メッセージ）を参照してください。

例　題　オクターブ4のCの音（ド）を、音量64で鳴らします。

```
1000 OUTM &H90,60,64
```

関連命令　PLAY：音楽の演奏を行います。

O

PAD関数

パッド関数　*pad*

パッド

```
PAD ⎧ 1 ⎫
    ⎩ 2 ⎭
```

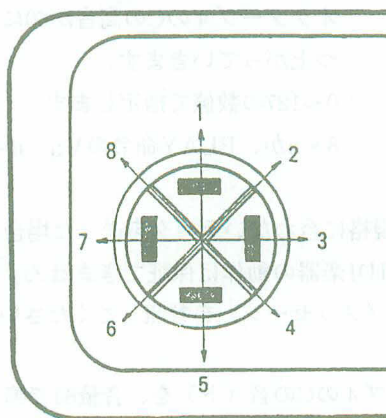
機　能　パッドが向いている方向を返します。

説　明　<1>、<2>……<1>：向かって左側のポートに接続されたパッドの方向を返します。
<2>：向かって右側のポートに接続されたパッドの方向を返します。

●どの方向にも向いていない時は、0を返します。
●返された復帰情報1～8は、パッドが下の図の方向を向いていることを示します。



RUNまたはセレクトボタンが押されているときは、PAD関数は正しい値を返しません。



関連命令　PTRIG関数：パッドのボタンの状態を返します。

PAINT

ペイント *paint*　　　　画面表示

1

```
PAINT ｛(wx, wy)      ｝｛ , ｛色            ｝｝〔, 境界色〕…
      ｛STEP (x, y)   ｝｛   ｛タイルストリング ｝｝
```

機　能　色を塗ります。

説　明　<境界色>で囲まれた範囲を、<色>で塗りつぶします。

<(wx, wy)>……塗り始めるドットの位置をワールド座標で指定します。

<STEP (X, Y)>……塗り始めるドットの位置を最終参照座標（LP）からの距離（x, y）で指定します。

<色>……塗りつぶす色を指定します。
（☞「●色の指定」p.54参照）
省略すると、直前のCOLOR命令の<前景色>で塗りつぶされます。

<タイルストリング>……塗りつぶすタイルパターンを文字式または16進数で指定します。（☞「●パターンの指定」p.61参照）

<境界色>……塗りつぶす範囲を囲む境界の色を、8つまで指定します。
（☞「●色の指定」p.54参照）
省略すると、<色>と同じ色を指定したとみなされます。

P

2 PAINT @ {(wx, wy) | STEP (x, y)} [, {色 | タイルストリング}]

機　能　色を塗り直します。

説　明　指定した位置のドットと同じ色でつながっている領域を、<色>で塗りつぶします。

<(wx, wy)>　……塗り始めるドットの位置をワールド座標で指定します。

<STEP (x, y)>　……塗りはじめるドットの位置を最終参照座標（LP）からの距離（x, y）で指定します。

<色>　……塗りつぶす色を指定します。
（☞「●色の指定」p. 54参照）
省略すると、直前のCOLOR命令の<前景色>で塗りつぶされます。

<タイルストリング>……塗りつぶすタイルパターンを指定します。
（☞「●パターンの指定」p. 61参照）

例　題　箱を表示し、その箱を塗りつぶします。

```
1000  CLS
1100  X=400:Y=200:D=100
1200  A1X=X:A1Y=Y:B1X=A1X+D:B1Y=A1Y+D
1300  A2X=A1X+D+10:A2Y=Y:B2X=A2X+D:B2Y=A2Y+D
1400  A3X=A2X+50:A3Y=A2Y+50:B3X=B2X-10:B3Y=B2Y-10
1500  LINE (A1X,A1Y)-(B1X,B1Y),PSET,6,B
1600  LINE (A2X,A2Y)-(B2X,B2Y),PSET,5,B
1700  LINE (A3X,A3Y)-(B3X,B3Y),PSET,3,B
1800  PRINT "HIT ANY KEY" :A$=INPUT$(1)
1900  PAINT  (A1X+1,A1Y+1),1,6              ' 四角形を塗りつぶす.
2000  PAINT  (A2X+1,A2Y+1),&H11111111,3,5         ' 模様で塗りつぶす.
2100  PAINT  (A3X+1,A3Y+1),&H01010101,3           ' 模様で塗りつぶす.
2200  END
```

関連命令　COLOR：テキスト画面またはグラフィック画面の、表示色や背景色を指定します。

PALETTE

パレット　*palette*

画面表示

```
PALETTE ［パレット番号, ｛色番号                  ｝］［, ｛0｝］
                       ｛[G輝度, R輝度, B輝度]｝     ｛1｝
```

＊［G輝度、R輝度、B輝度］の［　］は省略せず、そのまま記述します。

機　能　パレットの色を設定します。

説　明　16色モードおよび256色モードのとき、＜パレット番号＞に色を設定します。設定は、＜色番号＞または緑（G）、赤（R）、青（B）の輝度で指定します。

＜パレット番号＞……画面モードにより以下の数値で指定します。

16色モード　：0～15

256色モード：0～255

＜色番号＞　　……色を0～7の数値で指定します。

（☞「●色の指定」p. 54参照）

＜G輝度＞＜R輝度＞＜B輝度＞　……緑、赤、青の輝度を0～255の数値で個別に指定します。（☞「●色の指定」p. 54参照）

＜0＞＜1＞　　……1を指定すると、画面に切り替えノイズが出ないように内部でタイミングを取ってから、パレットを切り替えます。その分PALETTE命令の処理が遅くなります。
0を指定すると、タイミングはとりません。
省略すると、0を指定したとみなされます。

●オペランド（＜パレット番号＞、＜色番号＞、＜[G輝度、R輝度、B輝度]＞および0, 1）を全て省略したPALETTE命令を実行すると、パレットは初期設定と同じ状態になります。

●グラフィック２画面モードの場合、前側のページのパレット番号０番の色は常に透過色になり、変更できません。

例　題　四角形を表示し、色を変えていきます。

```
1000 SCREEN@ 0;CLS
1100 LINE(500,150)-(600,350),PSET,%15,BF
1200 '
1300 DIM G(15),R(15),B(15)
1400 FOR I=0 TO 15
1500   READ G(I),R(I),B(I)
1600 NEXT I
1700 '
1800 FOR I=0 TO 15
1900   FOR T=1 TO 5000:NEXT T
2000   PALETTE 15,[ G(I),R(I),B(I)]
2100   PRINT "PALETTE 15,[ ";G(I);",";R(I);",";B(I);"]"
2200 NEXT I
2300 END
2400 '
2500 DATA   0,  0, 0,  64, 64, 64,   0,  0,128,   0,  0,255
2600 DATA   0,128, 0,   0,255, 0,   0,128,128,   0,255,255
2700 DATA 128,  0, 0, 255,  0, 0, 128,  0,128, 255,  0,255
2800 DATA 128,128, 0, 255,255, 0, 128,128,128, 255,255,255
```

関連命令　COLOR：テキスト画面またはグラフィック画面の、表示色や背景色を指定します。

P

PART

パート　*part*

音楽/音声

PART　パート，チャンネル

機　能　PLAY命令で指定するパートをチャンネルに割り当てます。

説　明　<パート>　……パートの番号を0～15の数値で指定します。

<チャンネル>……演奏チャンネルを0～31で指定します。
演奏チャンネルの番号は次のように割り当てられています。

チャンネル番号	演奏チャンネル
0～5	内蔵FM音源チャンネル1～6
6～13	内蔵PCM音源チャンネル1～8
14～15	ダミー
16～23	MIDIポートAのチャンネル1～8
24～31	MIDIポートBのチャンネル1～8

- PLAY命令の"MML"はパートの番号順に記述され、<チャンネル>と対応付けられます。すなわち、PLAY命令のｎ番目のMMLは、PART命令でパート（n−1）に設定したチャンネルから演奏されます。
- PART命令を実行しない限り、パート0～15はそれぞれチャンネル0～15に割り当てられています。

関連命令　PART関数：パートに割り当てられているチャンネル番号を返します。
OUTM　　：MIDIポートへデータを送ります。

PART関数

パート関数　*part*

音楽/音声

```
PART (パート)
```

機　能　パートに割り当てられている演奏チャンネル番号を返します。

説　明　<パート>……0～15の数値で指定します。

●<パート>に割り当てられている演奏チャンネル番号を、0～31のいずれかの値で返します。

関連命令　PART：パートに演奏チャンネルを割り当てます。

P

PASTEL

パステル　*pastel*

PASTEL〔比率〕

機　能　混合比率を指定します。

説　明　描画位置の表示色と描画色とを混合するときの混色比率を指定します。

<比率>……混色比率を0～256の数値で指定します。
0を指定すると、現在画面に表示されている色になります。
256を指定すると、グラフィック命令で指定した<色>そのもので描かれます。
128を指定すると、もともと表示されている色と指定色を半々に混合します。
省略すると、128を指定したとみなされます。

- グラフィック命令中の論理操作でPASTELを指定するときは、PASTEL命令で混合比率を設定しておきます。
- BASIC起動時は、PASTEL 128が設定されています。

例　題　四角形を描く前に混合比率を設定します。

```
10 SCREEN@ 1
20 PASTEL 100
30 LINE(0,0)-(100,100),PASTEL,6,BF
40 LINE(50,50)-(150,150),PASTEL,5,BF
50 LINE(100,100)-(200,200),PASTEL,2,BF
60 END
```

PCMPLAY

ピーシーエム・プレイ　*PCM play*　　音楽/音声

```
PCMPLAY　配列名〔, 音量〕
```

機　能　PCM音声データをPCM音源に送って発声させます。

説　明　<配列名>にある音声データを、演奏チャンネルに送って発声させます。

<配列名>……音声データを読み込んだ配列を指定します。

<音量>　……1～127の数値で指定します。
1が最小、127が最大の音量を示します。
省略すると、64を指定したとみなされます。

- 拡張子が“.SND”のPCM音声ファイルをLOAD@命令で配列に取り込み、<配列名>でその配列を指定して、発声させることができます。



PCMPLAY命令を実行すると、PCM音源の専用メモリ（waveRAM）の内容が壊されるため、その後PCM音色データを演奏しようとしてもPCMチャンネルの音は鳴りません。
ファイル名を省略したLOAD@命令を実行して、PCM音色データをCD-ROMから読み直す必要があります。

例　題　PCMREC命令の例題を参照してください。

関連命令

PCMREC　：マイクからの音声をサンプリングして、データを配列に読み込みます。
LOAD@　：PCM音源またはFM音源用のデータをファイルから読み込みます。
VOICE SET：配列の音声データを音色データとして設定します。
PLAY　：音楽の演奏を行います。

P

PCMREC

ピーシーエム・レック　*PCM record*　　音楽/音声

PCMREC　配列名，サンプリング周波数

機　能　マイクからの音声をサンプリングして、データを配列に入れます。

説　明　<配列名>……DIM命令で配列宣言した配列を指定します。
配列には、（<サンプリング周波数>×録音秒数＋32）バイトの大きさが必要です。
文字型の配列は指定できません。

<サンプリング周波数>……19200以下の整数を指定します。
この値が大きくなるほど音質は良くなります。
8000を指定すると、電話程度の音質で録音されます。

例　題　マウスを左クリックしてから，内蔵マイクに音声を録音します。
もう一度マウスを左クリックすると，録音した音声が再生されます。

```
1100 REM   サンプリング録音==>再生
1200 COLOR 7,0,0 : SCREEN@ 0 : WIDTH 80,25 : CLS
1300 DIM A%(20000)
1400 MOUSE 0 : MOUSE 1,0,0,0 : WHILE MOUSE(2,0)=0 : WEND : MOUSE 5
1500 PCMREC A%,19200
1600 MOUSE 0 : MOUSE 1,0,0,0 : WHILE MOUSE(2,0)=0 : WEND : MOUSE 5
1700 PCMPLAY A%,127
1800 END
```

（このプログラムは2MBのメモリが必要です。）

関連命令　PCMPLAY：PCM音声データをPCM音源に送って発声させます。
LOAD@　：PCM音源またはFM音源用のデータをファイルから読み込みます。
VOICE SET：配列に読み込まれた音声データを、音色データとして設定します。
PLAY　：音楽の演奏を行います。

PEEK関数

ピーク関数　*peek*


```
PEEK（メモリアドレス〔，型番号〕）
```

機　能　メモリ上の指定した位置のデータを読み込みます。

説　明　<メモリアドレス>……読み込むデータの位置を、アドレスで指定します。
メモリアドレスは、[セレクタ]オフセットで指定し、<[セレクタ]>を省略した場合、BASICで使用するデータセグメント内のオフセットを指定したとみなされます。
<オフセット>は省略できません。
BASICのデータセグメント内では、VARPTR関数によって得られる変数のアドレスを指定します。

<型番号>……読み込みの単位を1、2または4で指定します。それぞれの数値は、以下の意味を持っています。
1：BYTE（1 byte）単位で読み込みを行います。
2：WORD（2 byte）単位で読み込みを行います。
4：DWORD（4 byte）単位で読み込みを行います。
省略すると1を指定したとみなされます。

●<メモリアドレス>は、範囲外を指定すると、メモリアドレス指定エラーとなります。
●得られる値は以下のようになります。

型番号　1の場合：0～255
　　　　2の場合：0～65535
　　　　4の場合：−2147483648&～2147483647&

●セレクタを（　）で指定しているV1.1 L20以前のプログラムを、V1.1 L21以降のBASICで実行すると正しく動作しません。プログラムを変更してから実行してください。

●<(セレクタ)>には、次の値が指定できます。

セレクタ値	許されるオフセット値	アクセス対象
1Ch 104h 120h	0~7FFFFh	VRAM (32768色モード、16色モード)
10Ch 128h	0~7FFFFh	VRAM (256色モード)
114h 130h	0~1FFFFh	スプライトパターンRAM
14h	不定*	BASIC変数、配列など

*:指定できるオフセットの範囲は不定のため、PEEK/POKEで参照する直前に、VARPTR関数で得たアドレスを指定してください。

●VARPTR関数が返すアドレスには、データが次のような形式で格納されています。

・整数型の場合(&H0123)

+0	+1 バイト
23	01

・ロング型整数の場合(&H01234567)

+0	+1	+2	+3 バイト
67	45	23	01

・文字型の場合(" ABCDE ")

+0	+1	+2	+3	+4	+5 バイト
67	45	23	01	23	01

―オフセット― ―長さ―

オフセットは、文字列の格納されているアドレスをBASIC用データセグメント（セレクタ14h）内のオフセット値で示しています。
長さは、文字列の長さをバイト数で示しています。
即ちオフセット&H01234567に、&H0123バイトの文字列が格納されています。

オフセット	+0	+1	+2	+3	+4
	A	B	C	D	E

・単精度実数型の場合

+0 ～ +22	+23 ～ +30	+31ビット
仮数部	指数部	符号

・倍精度実数型の場合

+0 ～ +51	+52 ～ +62	+63ビット
仮数部	指数部	符号

関連命令　POKE　：メモリ上の指定した位置にデータを書き込みます。
VARPTR：変数のアドレスを返します。

PLAY

プレイ *play*

音楽/音声

```
PLAY "MML" 〔,"MML"〕……
```

機　能　音楽演奏を行います。

説　明　各パートの<MML>を同時に演奏チャンネルに送って演奏します。

<MML>……テンポ、音長、音高、音量などの設定をMML（Music Macro Language）で指定します。（「MML」p.90参照）

- 16和音までの演奏ができます。
  1個目のMMLがパート0、2個目のMMLがパート1、……16個目のMMLがパート15というように、16個のパートまで指定できます。
- <MML>を指定するパートの前に、指定を行わないパートがある場合は、その数だけコンマ（,）を記述しなければなりません。
- PART命令で演奏チャンネルの割り当てを変更しない限り、パート0~5までが本体内蔵FM音源のチャンネル1~6に、パート6~13までが本体内蔵PCM音源のチャンネル1~8に、それぞれ対応します。なお、パート14と15には、音源が割り当てられていません。



PCMPLAY命令を実行した後は、waveRAMの内容が壊されるため、その後PLAY命令でPCM音色データを使って演奏させようとしても、PCMチャンネルの音は鳴りません。
ファイル名を省略したLOAD@命令を実行して、PCM音色データをCD-ROMから読み直す必要があります。

関連命令　PLAY関数：PLAY文の演奏中かどうかの状態を返します。

PLAY@

プレイ・アットマーク　*play@*

音楽／音声

```
PLAY@  配列名 [, {0 | 1}]
```

機　能　EUPファイルを演奏します。

説　明　拡張子が".EUP"のファイルをLOAD@命令で配列に取り込んでおくと、配列名を指定して演奏させることができます。

<配列名>　……EUPのデータを読み込んだ配列を指定します。

<0>、<1>……ループ演奏させるかどうかを指定します。
0：ループ演奏しない。
1：ループ演奏する。
省略すると、0を指定したとみなされます。

- 演奏中の配列を書き換えたり、読み込んだりすると、演奏が止まったり、正しく演奏されなかったりすることがあります。
- V1.1のBASICでは、PLAY@命令は使用できません。

例　題　EUPファイル" SAMPLE.EUP"を読み込んで演奏します。

```
1000 CLEAR
1100 PLAY OFF                ' 音楽機能の初期化
1200 DIM A%(30000)           ' ＥＵＰファイルを読み込む配列を確保する
1300 LOAD@ "SAMPLE.EUP",A%   ' ＥＵＰファイルを配列に読み込む
1400 PLAY@ A%                ' 配列を演奏する
1500 END
```

関連命令　LOAD@　　：EUPファイルを配列に読み込みます。

P

PLAY関数

プレイ関数　*play*

音楽/音声

1
```
PLAY(0)
```

機　能　演奏中かどうかの状態を返します。

説　明　PLAY命令による音楽の演奏中ならば−1を、演奏中でなければ0を返します。

2
```
PLAY(1)
```

機　能　PLAY命令で演奏中の小節番号を返します。

説　明　演奏を始めた最初の小節を1としたときの現在演奏中の小節番号を返します。
PLAY命令の演奏が途中で終わると、小節番号はそこから数え直されます。
演奏中でない場合、返される値は不定です。



4／4拍子以外の曲の場合は、MMLの拍子の宣言（%Sn／m）で宣言をしておかないとPLAY（1）は正しい値を返しません。

3 PLAY(2)

機　能　PCM音源が再生中かどうかを返します。

説　明　PCM音源が、PCMPLAY命令により再生中ならば-1を、再生中でなければ0を返します。

関連命令　PLAY　　　：音楽の演奏を行います。
PCMPLAY：PCM音声データをPCM音源に送って発声させます。

P

PLAY ON/OFF/STOP

プレイ・オン/オフ/ストップ *play on/off/stop*

音楽/音声

```
PLAY { ON   }
     { OFF  }
     { STOP }
```

機　能　PLAY命令による演奏を許可、中止（初期化）または一時停止します。

説　明　●PLAY ON

演奏を許可します。演奏が中断していた場合には、再開します。

●PLAY OFF

演奏を中止し、"MML"の内容をすべて消去して、音楽機能を初期化します。ただし、命令の実行がPLAY命令実行後か、PLAY＠命令実行後かによって初期化の内容が違います。

初期化の項目	PLAY実行後の PLAY OFF	PLAY＠実行後の PLAY OFF
"MML"の内容をすべて消去	○	○
MML演奏の各パートのオクターブ	○	○
MML演奏の各パートの音長	○	○
MML演奏の各パートの音量	○	○
MML演奏の各パートのアクセント	○	○
MML演奏の各パートの1音中の音長の割合	○	○
MML演奏のテンポ	○	○
MML演奏の拍子	○	○
各トラックの出力ポート	○	○
各トラックの出力MIDIチャンネル	○	○
各トラックのミュート		○
各音源のMIDIチャンネル		○
各音源の各チャンネルの音量と定位		○
EUP演奏のループの解除		○

●PLAY STOP

演奏を中断しますが、PLAY　ON命令が実行されると、演奏を再開します。PLAY STOP後、新たにPLAY命令でMML データを送ることもできます。PLAY命令でデータを送ったために、バッファがオーバーフローしたときは、自動的にPLAY ONの状態になります。

例　題　演奏を一時停止した後、再度演奏を再開させます。

```
1000  PLAY "L4O4"
1100  PLAY "CDEFGAB>CR4C<BAGFEDC"
1200  FOR I=1 TO 4500:NEXT I
1300  PLAY STOP                    ' 演奏を一時停止する.
1400  A$=INKEY$:IF A$="" THEN 1400
1500  PLAY ON                      ' 演奏を再開する.
1600  END
```

関連命令　PLAY：音楽の演奏を行います。

PLAY@：EUPファイルを演奏します。

P

POINT

ポイント *point*　　　　画面表示

```
POINT  { (wx, wy)
       { STEP (x, y)
```

機　能　最終参照座標（LP）を変更します。

説　明　グラフィック機能においては、最後に参照した座標（LP）が常に記憶されています。LPは、相対座標で座標指定するときの基点になります。
POINT命令は、このLPの値を強制的に変更します。

＜wx, wy＞　……新LPをワールド座標値で指定します。

＜STEP（x, y）＞……新LPを現在のLPからの距離（x, y）で指定します。

例　題　LPの位置を変更して図を表示します。

```
1000  CLS
1100  X=150:Y=200:D=30
1200  FOR N=1 TO 3
1300    POINT (X,Y)          ' LP を変更する
1400    GOSUB 1700           ' print
1500  NEXT N
1600  END
1700  '---print---
1800  WY=Y
1900  FOR I=1 TO 5
2000    X=X+D:WY=WY+D
2100    LINE -(X,WY),PSET,7,B
2200  NEXT I
2300  RETURN
```

関連命令　POINT関数（形式2）：最終参照座標（LP）を返します。

POINT関数

ポイント関数　*point*　　画面表示

1

```
POINT(sx, sy)
```

機　能　指定した位置にドットが描かれているかどうかを返します。

説　明　<sx, sy>……スクリーン座標値で位置を指定します。

●指定した位置にドットが描かれているかいないかによって−1か0を返します。

−1：ドットが描かれている。
（背景色以外の色で表示されている）

0：ドットが描かれていない。
（背景色で表示されている）

●<sx, sy>がVRAMの範囲外のときは0を返します。

例　題　指定した位置にドットが設定されているかいないかを調べます。

```
1000 PSET(320,100),7
1100 A=POINT(320,100)          '調べたい位置を指定する.
1200 B=POINT(321,100)
1300 PRINT A,B
1400 END

結果：-1        0
```

P

2 POINT(機能)

機　能　最終参照座標（LP）を返します。

説　明　グラフィック命令で最後に参照された座標（LP）を、ワールド座標値、またはスクリーン座標値で返します。

<機能>……どの座標値を返すかを0～3の数値で指定します。

0：LPのx座標がワールド座標値で返されます。
1：LPのy座標がワールド座標値で返されます。
2：LPのx座標がスクリーン座標値で返されます。
3：LPのy座標がスクリーン座標値で返されます。

例　題　長方形を描き、そのときのLPの値を表示します。

```
1000 WAX=WINDOW(0):WAY=WINDOW(1)
1100 WBX=WINDOW(2):WBY=WINDOW(3)
1200 VAX=VIEW(0):VAY=VIEW(1)
1300 VBX=VIEW(2):VBY=VIEW(3)
1400 '
1500 WINDOW(-320,-200)-(319,199)
1600 VIEW(0,0)-(639,399)
1700 LINE(100,0)-(250,150),PSET,3,B
1800 PRINT "POINT(0)";POINT(0)
1900 PRINT "POINT(1)";POINT(1)
2000 PRINT "POINT(2)";POINT(2)
2100 PRINT "POINT(3)";POINT(3)
2200 '
2300 WINDOW(WAX,WAY)-(WBX,WBY)
2400 VIEW(VAX,VAY)-(VBX,VBY)
2500 END

結果：POINT(0)   250
      POINT(1)   150
      POINT(2).  570
      POINT(3)   350
```

関連命令　POINT：最終参照座標（LP）を変更します。

POKE

ポーク *poke*


```
POKE メモリアドレス，書き込む値［，型番号］
```

機　能　メモリ上の指定した位置にデータを書き込みます。

説　明　＜メモリアドレス＞……書き込むデータの位置を、アドレスで指定します。
メモリアドレスは、[セレクタ]オフセットで指定し、＜[セレクタ]＞を省略し場合、BASICで使用するデータセグメント内のオフセットを指定したと見なされます。
＜オフセット＞の省略はできません。BASICのデータセグメント内では、VARPTR関数によって得られる変数のアドレスを指定します。
範囲外を指定すると、エラーになります。

＜書き込む値＞　……書き込むデータを以下の範囲で記述します。

型番号　1の場合：0～255
　　　　2の場合：−32768%～32767%
　　　　4の場合：−2147483648&～2147483647&

＜型番号＞　……書き込みの単位を1、2または4で指定します。それぞれの数値は、以下の意味を持っています。

1：BYTE（1 byte）単位で書き込みを行います。
2：WORD（2 byte）単位で書き込みを行います。
4：DWORD（4 byte）単位で書き込みを行います。

省略すると、1を指定したとみなされます。

- セレクタ値として指定できる値および変数の内部格納形式については、PEEK関数を参照してください。
- セレクタを（　）で指定しているV1.1 L20以前のプログラムを、V1.1 L21以降のBASICで実行すると正しく動作しません。プログラムを変更してから実行してください。

関連命令　PEEK関数　　：メモリ上の指定した位置のデータを読み込みます。
VARPTR関数：変数のアドレスを返します。

P

POS関数

ポジション　*position*

画面表示・印刷

POS（ファイル番号）

機　能　テキスト画面上のカーソルの水平位置またはプリンタバッファ内の水平文字位置を返します。

説　明　<ファイル番号>……0またはプリンタ装置に割り当てられたファイル番号を数値で指定します。
0を指定するとテキスト画面上のカーソルの水平位置を返します。
プリンタのファイル番号を指定するとプリンタバッファ内の水平文字位置を返します。

例　題　画面上のカーソルの水平位置、垂直位置を画面に表示します。

```
1000  LOCATE 10,3
1100  A=CSRLIN
1200  B=POS(0)                          'カーソルの水平位置をBに返す.
1300  PRINT A,B
1400  END

結果：     3          10
```

関連命令　LPOS関数　：プリンタバッファ内の水平文字位置を返します。
CSRLIN関数：画面上のカーソルの垂直位置を返します。

PRINT〔省略形?〕

プリント　*print*

画面表示

```
PRINT  ［式［｛ , ｝〔式〕］…］
             ｛ ; ｝
```

機　能　式の結果を画面に表示します。

説　明　<式>……数値式または文字式を指定します。

複数個の<式>を記述する場合は、コンマ（,）またはセミコロン（;）で区切ります。

<式>を全て省略すると、改行のみを行います。

- セミコロン（;）直後の<式>の結果は、前の式の結果に続いて表示します。
  式の結果が数値の場合は、数値の前後に1個ずつの空白が取られます。前の空白は符号を表示するための領域で、数値が負のときはマイナス（−）を、正のときは空白が入ります。
- コンマ（,）直後の<式>の結果は、前の<式>の結果が表示されているフィールドの次のフィールドの先頭から表示します。
  式の結果が2つ以上のフィールドにまたがる場合、後の<式>の結果は前の<式>の結果のフィールドの次のフィールドの先頭から表示されます。



フィールド

1行は半角14文字の単位で区切られています。この単位をフィールドといいます。

- 記述がコンマ（,）またはセミコロン（;）で終わっている場合は、結果を表示した後に改行しません。次のPRINT命令による表示が、同じ行に表示されます。
  コンマ（,）またはセミコロン(;)で終わっていない場合は、自動的に改行します。
- 結果の長さが画面の表示幅より大きい場合は、行の終わりまで表示した後、残りを改行後に表示します。
- 結果の長さが、現在のカーソル位置から行の終わりまでの長さより大きい場合は、

P

改行してから次の行に表示します。

例　題　表示形式を変えて、文字列を表示します。

```
1000  A$="富士通株式会社":B$="ＢＡＳＩＣ"
1100  C%=1234:D%=5678:E%=-9876
1200  PRINT A$,B$            ' フィールドごとに表示する.
1300  PRINT A$;B$            ' 文字を続けて表示する.
1400  PRINT A$
1500  PRINT B$
1600  PRINT C%,D%
1700  PRINT C%;D%
1800  PRINT C%;E%
1900  END

結果：富士通株式会社              ＢＡＳＩＣ
      富士通株式会社ＢＡＳＩＣ
      富士通株式会社
      ＢＡＳＩＣ
       1234           5678
       1234  5678
       1234 -9876
```

関連命令　WRITE　：式の結果を画面に表示します。
WRITE#：式の結果を指定したファイルに出力します。
PRINT#：式の結果を指定したファイルに出力します。
LPRINT：式の結果をプリンタに出力します。

PRINT#〔省略形?#〕

* プリント・シャープ *print#*

印刷・データファイル

1	PRINT #ファイル番号〔, 式〔{, / ;}〔式〕〕…〕

機　能　式の結果を、指定したファイルに出力します。

説　明　<ファイル番号>……出力先のファイルのファイル番号を指定します。

<式>……数値式、または文字式を指定します。
<式>を複数記述する場合は、その間をコンマ（,）またはセミコロン（;）で区切ります。

- <ファイル番号>で指定したファイルは、出力モード（O）または追加モード（A）で既にオープンされている必要があります。
- 複数の<式>をコンマ（,）で区切って指定すると、PRINT命令と同様に14文字ずつのフィールドがとられ、各式の結果はフィールドの最初から出力します。
- 複数の式をセミコロン（;）で区切って指定すると、各<式>を直前の<式>に続いて出力します。
- ディスクに文字列を出力するときは、データの直後に区切り記号のダブルクォーテーション（"）を記述する必要があります。区切り記号を使用せずに複数の文字列データを指定すると、1つの文字列として出力されます。したがって、そのデータをINPUT命令で読み込むと、連続した1つの文字列となります。
- ダブルクォーテーション（"）を出力する場合は、CHR$（&H22）を用います。

P

例　題　ファイル名“DATA2DAT”に、キーボードから入力したデータを書き込みます。

```
1000 OPEN"O",#1,"DATA2DAT"
1100 INPUT "品名：",HINMEI$
1200 IF HINMEI$="END"THEN 1700
1300 INPUT "金額：",KINGAKU
1400 PRINT#1,HINMEI$,",",KINGAKU
1500 GOTO 1100
1600 CLOSE#1
1700 END

結果：品名：にんじん
　　　金額：45
　　　品名：大根
　　　金額：100
　　　品名：いも
　　　金額：30
　　　品名：END
```

2　PRINT #ファイル番号, USING 書式文字列 ; 〔 式 〔 {, / ;} 〔式〕 〕 … 〕

機　能　式の結果を、指定した書式でファイルに出力します。

説　明　PRINT#命令の形式1と同様の処理を行います。異なる点は、＜USING 書式文字列＞で、出力時の書式を指定できるところです。

＜書式文字列＞……出力される文字や数値の形式（書式）を示す書式制御文字、または定数文字列を指定します。

（☞「書式制御文字」p. 72参照）

関連命令　WRITE#：式の結果を、ファイルに出力します。

PRINT USING〔省略形？USING〕

プリント・ユージング　*print using*　　　　画面表示

```
PRINT　USING　書式文字列；〔式〕{ , | ; }〔式〕…
```

機　能　指定した書式で式の結果を画面に出力します。

説　明　<書式文字列>……書式制御文字を使用して文字や数値の出力形式を指定します。
（「書式制御文字」p. 72参照）

<式>……画面に出力する文字列または数値を指定します。
複数個の<式>を記述する場合は、間をコンマ（,）またはセミコロン（;）で区切ります。

- 実際の数値が、書式制御文字で指定した桁数で表せないときは、数値の先頭にパーセント記号（%）が出力されます。
数値の丸めの結果、指定桁数を超えた場合でも、丸められた数値の前にパーセント記号（%）が出力されます。
- 記述がコンマ（，）またはセミコロン（；）で終わっている場合は、結果を表示した後に改行しません。次のPRINT USING命令による表示が、同じ行に表示されます。コンマ（，）またはセミコロン（；）で終わっていない場合は、自動的に改行します。

P

例　題　書式文字列を変えて、数値または文字列を表示します。

```
1000  A=12345:B=62.55:C=-160:D=3.251E+12
1100  PRINT USING "########  ######.#";A;A
1200  PRINT USING "+#######  #######-";B;C
1300  PRINT USING "**#.###  ¥¥######";B;C
1400  PRINT USING "**¥#####  **¥####,";B;A
1500  PRINT USING "#####^^^^ ";D
1600  A$="ABCDE":B$="FGHIJKL"
1700  C$="BASIC"
1800  PRINT USING "&        &---&     &---!";A$;B$;C$
1900  END
```

結果

```
     12345   12345.0
       +63      160-
   **62.550    -¥160
   *****¥63 *¥12,345
    32510E+08
   ABCDE     ---FGHIJKL---B
```

関連命令　LPRINT USING：指定した書式で文字または数値を、プリンタに出力します。
PRINT　　　　　：式の結果を画面に表示します。
PRINT#　　　　 ：式の結果をファイルに出力します。

PSET/PRESET

ピーセット/ピーリセット　*point set/point reset*　　画面表示

```
PSET    { (wx, wy)    }  〔, 〔色〕〔, 論理操作〕〕
        { STEP (x, y) }

PRESET  { (wx, wy)    }
        { STEP (x, y) }
```

機　能　ドットを、<色>または背景色で表示します。

説　明　PSET命令は、ドットを指定した<色>で表示します。
PRESET命令は、ドットを背景色で表示します。

<wx, wy>　……表示するドットをワールド座標値で指定します。

<STEP（x, y）>　……表示するドットを、最終参照座標（LP）からの距離（x, y）で指定します。

<色>　……表示するドットの色を指定します。
（☞「●色の指定」p. 54参照）
省略すると、直前のCOLOR命令で指定した前景色で表示されます。

<論理操作>　……AND、OR、XORまたはPASTELのいずれかを指定します。
指定した<色>と、現在表示されているドットの色とを論理演算し、その結果の色で表示します。
省略すると、PSETを指定したとみなされます。

P

例　題　四角形を描き、対角線を背景色で引きます。

```
1000  CLS
1100  LINE (0,0)-(100,100),PSET,5,BF
1200  FOR I=0 TO 100
1300    PRESET (I,I)
1400  NEXT
1500  END
```

関連命令　LINE　：線または四角形を描きます。
PAINT：色を塗ります。

PTRIG関数

パッドトリガ関数　*pad trigger*

パッド

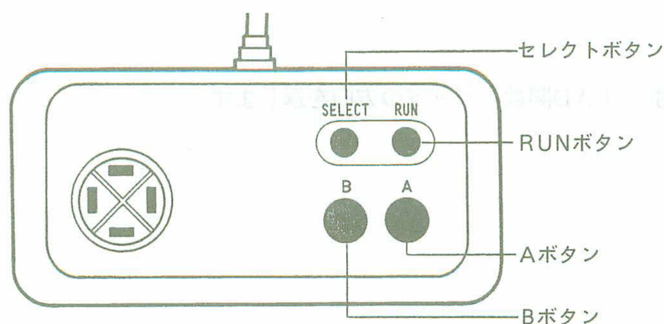
PTRIG(パッドポート番号)

機　能　パッドのボタンの状態を返します。

説　明　<パッドポート番号>で指定したパッドのセレクトボタン、RUNボタン、Aボタンおよびボタンの状態を返します。

<パッドポート番号>……パッドは2つまで接続でき、向かって左側のパッドを1、右側のパッドを2で指定します。

●パッドの各ボタンは、以下のとおりです。



P

●復帰情報の値と、各ボタンの関係は以下の一覧表のとおりです。

復帰情報	セレクト	RUN	B	A
0				
1				○
2			○	
3			○	○
4		○		
5		○		○
6		○	○	
7		○	○	○
8	○			
9	○			○
10	○		○	
11	○		○	○

（○はそのボタンが押されていることを示します。）

関連命令　PAD関数：パッドの方向を返します。

PUT

プット　*put*　　　　画面表示・データファイル

```
PUT〔#〕ファイル番号〔, レコード番号〕
```

機　能　バッファの内容を、ランダムファイルに出力します。

説　明　バッファの内容を、<ファイル番号>で指定したランダムファイルの、<レコード番号>で指定したレコードに書き込みます。

<#>　　　　　……省略しても意味は変わりません。

<ファイル番号>……オープン時にランダムファイルに割り当てられたファイル番号を指定します。

<レコード番号>……ランダムファイルの何番目のレコードに出力するかを1以上の値で指定します。
省略すると、直前にファイルにGETまたはPUTしたレコードの次のレコードに書き込まれます。

- <ファイル番号>で指定したファイルは、ランダム入出力モード（R）でオープンされている必要があります。
- <レコード番号>がファイルの最大レコードを超えていた場合は、ディスク上に新たな領域を確保して出力します。この場合は、新たな領域の確保のために十分な空領域が必要です。必要な空領域の大きさは、次の計算式で求められます。

((指定したレコード番号－現在の最大レコード番号）＊レコード長）バイト

- ランダムファイルのバッファには、LSET命令またはRSET命令でデータをセットします。
- ランダムファイルに数値データを書き込みたい場合は、MKI$／MKL$／MKS$／MKD$関数を使って文字型に変換してから書き込みます。

例　題　キーボードからデータを入力し、“DATA1DAT”というランダムファイルに出力します。ENDが入力されると、プログラムを終了します。

```
1000 OPEN "R",#1,"DATA1DAT"
1100 FIELD #1, 20 AS A$,12 AS B$
1200 LINE INPUT "名前は              ";NAM$
1300 IF NAM$="END" THEN 1900
1400 LINE INPUT "電話番号は          ";TEL$
1500 LSET A$=NAM$
1600 LSET B$=TEL$
1700 PUT #1
1800 GOTO 1200
1900 CLOSE #1 :END

結果：名前は     警察
      電話番号は 110
      名前は     消防
      電話番号は 119
      名前は     END
```

関連命令　GET　：ランダムファイルから、データをバッファに読み込みます。

LSET／RSET：文字データを左詰め、右詰めでランダムファイルのバッファのフィールドに代入します。

MKI$／MKL$／MKS$／MKD$関数：数値を文字列に変換します。

PUT@

プット・アットマーク　*put@*　　　　画面表示

1　PUT@ (sx1, sy1) − (sx2, sy2), 配列名〔,〔論理操作〕〔, 色〕〕

機　能　GET@命令（形式1）で配列に読み込まれたドットパターンを、画面に<色>で表示します。

説　明　<sx1, sy1>, <sx2, sy2>……表示するドットパターンの位置を、四角形の対角線上の2点のスクリーン座標で指定します。
スクリーン座標値は、VRAMの範囲を超えて指定することができます。

<配列名>……ドットパターンが読み込まれている配列を指定します。
配列内には、そのドットがオンかオフかの情報のみで、表示されるドットの色についての情報はありません。

<論理操作>……PSET、PRESET、AND、OR、XOR、NOT、PASTEL、OPAQUEのいずれかを指定します。
（☞「●グラフィック命令の論理操作」p. 60参照）
省略すると、PSETを指定したとみなされます。

<色>……表示するドットの色を指定します。
（☞「●色の指定」p. 54参照）
省略すると、直前のCOLOR命令で指定した前景色で表示されます。

2

PUT@A （sx 1，sy 1）－（sx 2，sy 2），配列名
〔，〔論理操作〕〔，〔横倍率〕〔，〔縦倍率〕〔，〔透過色〕〔，オフセット〕〕〕〕〕

機　能　GET＠A命令（形式2)で配列に読み込まれたグラフィックデータを、画面に表示します。

説　明　＜sx1, sy1＞＜sx2, sy2＞……表示するドットパターンの位置を、四角形の対角線上の２点のスクリーン座標で指定します。
これらは、VRAMの範囲を超えて指定できます。

＜配列名＞……グラフィックデータが読み込まれている配列を指定します。

＜論理操作＞……PSET、PRESET、AND、OR、XOR、NOT、PASTEL、MATTEのいずれかを指定します。
（☞「●グラフィック命令の論理操作」p. 60参照）
省略すると、PSETを指定したとみなされます。

＜横倍率＞＜縦倍率＞……GET＠A命令で読み込んだパターンを、拡大または縮小表示する場合に指定します。
省略すると、１を指定したとみなされます。
横または縦のドット数に＜倍率＞を掛けた値が16, 384未満である必要があります。

＜透過色＞……ドットの描画時に省く色を指定します。
＜論理操作＞でMATTEを指定したときは、必ず指定しなければなりません。
（☞「●色の指定」p. 54参照）

＜オフセット＞　……配列の途中から表示するとき、表示の開始位置を、（配列の先頭から数えた要素の個数）－1で指定します。多次元配列では、0～（各次元の要素数の積）－1の値が指定できます。
省略すると、0を指定したとみなされ、配列の先頭から表示されます。

●GET@A命令でグラフィックデータを読み込んだときと同じ画面モードで実行しないと、正しく表示されません。

例　題　ドットパターンを読み込んで、指定した場所に指定した色で表示します。

```
1000  CLS
1100  SYMBOL (150,50), "富士通", 4, 2, 1
1200  DIM A%(INT((INT((192+7)/8)*40+2-1)/2))
1300  GET@ (150,50)-(341,89),A%
1400  PUT@ (150,100)-(341,139),A%,,2   '指定した場所に赤で表示する
1500  END
```

関連命令　GET@　：グラフィック画面上のドットパターンを配列に読み込みます。
GET@A　：グラフィック画面上のドットパターンを、その色の情報と共に配列に読み込みます。

P

RANDOMIZE〔省略形RNDM.〕

ランダマイズ *randomize*　　　計算

```
RANDOMIZE 〔式〕
```

機　能　乱数の系列を変更します。

説　明　<式>……乱数系列の番号を、−2147483648～+2147483647の範囲で指定します。省略すると、プログラムの実行を停止し、「乱数の系列の番号を入力してください（−2147483648～+2147483647）?」というメッセージを表示します。画面の指示にしたがって数値を入力してください。入力後、実行が再開します。

●RANDOMIZE命令を使って乱数系列の変更を行わない場合、RND命令は実行されるたびに同じ系列の乱数を発生させます。

例　題　同じ系列の乱数5個を発生させ、表示します。
乱数の系列は、TIME関数でプログラムの実行時の時刻を指定しています。

```
1000 RANDOMIZE TIME
1100 FOR I=1 TO 5
1200   A=RND
1300   PRINT USING "#.#########";A
1400 NEXT I
1500 END

結果：0.443902000
      0.149444000
      0.940090000
      0.045970700
      0.131892000
```

関連命令　RND：0以上1未満の乱数を返します。

READ

リード　*read*　　　　　　　　　　　　　　　　　　　　　　データ入力

```
READ　変数名〔, 変数名〕…
```

機　能　DATA命令で定義した定数を、変数と１対１に対応させて読み込みます。

説　明　＜変数名＞……数値変数、または文字変数を指定します。

- ＜変数名＞の型は、DATA命令で定義した定数の型と一致している必要があります。
- 複数のREAD命令で定義された変数は、行番号の若い順に一連の連続した変数として扱われます。
- 複数のDATA命令で定義された定数は、行番号の若い順に一連の連続した定数として扱われます。
- RESTORE命令で、DATA命令の記述行を指定することにより、定数の読み込みを任意のDATA命令から始めることができます。

例　題　DATA命令で定義した定数を、READ命令で変数に読み込みます。D1には数値定数が、D2$には文字定数が読み込まれます。

```
1000 READ D1,D2$                    '1400 行のDATAを変数D1, D2に読み込む
1100 IF D1=0 THEN END
1200 PRINT D1,D2$                   ' 変数D1, D2$ を画面に出力する
1300 GOTO 1000
1400 DATA 1,a,2,bb,3,ccc,0,dddd
```

関連命令　DATA　　　：READ命令で読み込む数値定数、文字定数を定義します。
RESTORE：READ命令の読み込み対象となるDATA命令を指示します。

REM[省略形']

リマーク　*remark*

一般命令

```
REM　注釈文字列
```

機　能　プログラムの中に注釈を指定します。

説　明　<注釈文字列>……注釈を記述します。注釈文字列の中に何を記述してもプログラムの実行には影響しません。

- REM命令では、コロン（:）も注釈の一部とみなされます。したがって、1行をコロン（:）で区切っても全体が注釈行とみなされ、実行されません。

例　題　PRINT "富士通" は、注釈の指定がされているので、実行されません。

```
REM PRINT "富士通"
```

RENUM

リナンバー *renumber*

コマンド

RENUM〔新行番号〕〔,〔{旧行番号 | ラベル名}〕〔, 増分値〕〕

機　能　プログラムの行番号を付け替えます。

説　明　<旧行番号>または<ラベル名>で指定した行から、プログラムの最後の行までを、<新行番号>の行番号で始まり、<増分値>を増分量とした行番号に付け替えます。

<新行番号>　……付け替え開始位置に新たに付ける行番号を指定します。
省略すると、10を指定したとみなされます。

<旧行番号>　<ラベル名>　……付け替えを行う開始位置を指定します。
省略すると、プログラムの最初の行から付け替えを行います。
存在しない<行番号>を指定すると、指定した番号よりも大きく、かつ最も近い行番号から、付け替えを行います。
存在しない<ラベル名>を指定すると、エラーになります。

<増分値>　……<新行番号>以降の行番号の間隔を指定します。
省略すると、10を指定したとみなされます。

● プログラムの中でRENUM命令を実行した場合は、RENUM命令の実行後、直ちにプログラムを終了し、BASICエディタに戻ります。
● RENUM命令の実行範囲内にGOTO、GOSUB、ON～GOTO、ON～GOSUB、ERL、IF～THENなどの命令が記述してある場合は、これらの命令中で参照している行番号も、新しい行番号に変更します。
参照している行番号が、プログラム内に存在しない場合はエラーとなり、行番号

R

は変更されません。

● RENUM命令により、65529を超える行番号を付けることはできません。また、RENUM命令により行番号を付け替えることによって、実行順序が変わるような指定もできません。(たとえば、行番号10, 20, 30の3つの行があるとき、30を15に付け替えるなど)

● RENUM命令は、割り込み処理ルーチン、エラー処理ルーチンの中では使用できません。

● BASICエディタの「その他」の「Renum」を指定した場合と同じ結果になります。

コンパイラ ● コンパイラでは、RENUM命令は使用できません。RENUM命令を含むプログラムをコンパイルするとエラーになります。

例題

```
1000 INPUT A$
1001 IF A$=" " THEN 1000
1200 PRINT A$
1400 END

RENUM 10,10                  ' 行番号を初期値10、増分値10で付け直す
LIST

結果：10 INPUT A$
      20 IF A$=" " THEN 1000
      30 PRINT A$
      40 END
```

RESTORE

リストア *restore*


```
RESTORE ［｛行番号　｝］
         ［｛ラベル名｝］
```

機　能　READ命令の読み込み対象になるDATA命令の位置を指定します。

説　明　<行番号> <ラベル名>……RESTORE命令後のREAD命令が読み込みを始めるDATA命令記述行を指定します。
省略すると、RESTORE命令後のREAD命令は、プログラムの最初のDATA命令から読み込みを始めます。

例　題　RESTORE命令でDATA命令の読み込み順を変更します。DATA命令の読み込みは、行番号1800、1600、1700の順になります。

```
1000 RESTORE 1800           '1800 行目を指示
1100 READ A$                '1800 行目のDATA命令よりデータを読み始める
1200 IF A$="" THEN RESTORE  ' 空文字があれば最初のDATA命令を指定
1300 IF A$="END" THEN END   ' "END" があれば, 終了
1400 PRINT A$
1500 GOTO 1100
1600 DATA H,I,J,K,L,M,N
1700 DATA O,P,Q,R,S,T,U,END
1800 DATA A,B,C,D,E,F,G,,

結果：ABCDEFGHIJKLMNOPQRSTU
```

関連命令　READ：DATA命令により定義された定数を、変数に読み込みます。
DATA：READ命令で読み込む数値および文字定数を定義します。

R

RESUME

リジューム　*resume*

エラー処理

```
RESUME  [ { NEXT    } ]
          { 行番号  }
          { ラベル名 }
```

機　能　エラー処理ルーチンから指定した行へ戻り、プログラムの実行を再開します。

説　明　<NEXT>……エラーの発生した次の行からプログラムを再開する場合に指定します。

<行番号><ラベル名>……プログラムを再開する行を指定します。

●プログラムを再開する行の指定を省略した場合は、エラーの発生した行から実行を再開します。この場合、RESUME命令の実行前に、エラーの原因を取り除いておかないと、再び同じエラーが発生し、エラー処理ルーチンに戻ってしまいます。

例　題　ON ERROR GOTO命令の例を参照してください。

関連命令

ERR／ERL　　　　：発生したエラーのエラー番号および行番号を返します。

ON ERROR GOTO：エラー処理ルーチンを定義し、エラートラップを許可します。

R

RETURN〔省略形RET.〕

リターン *return*

プログラムの分岐

```
RETURN [ { 行番号   } ]
         { ラベル名 }
```

機　能　サブルーチンを終了し、サブルーチンを呼び出したプログラムへ復帰します。

説　明　<行番号> <ラベル名>……復帰先の行を指定します。
省略すると、サブルーチンの実行を指示したGOSUBまたはON～GOSUB命令の直後の命令へ復帰します。

- 割り込み処理ルーチン内のRETURN命令では、復帰先の行指定を省略すると中断した箇所へ復帰します。

R

例　題　サブルーチンSUB1、SUB2または、SUB3を実行し、各々を呼び出したプログラムに復帰します。入力した値が変数Aまたは、Bの判定結果と一致するまでメッセージを表示して、次の入力を待ちます。一致すると“OK!”と表示してプログラムを終了します。

```
1000 A=-10: B=-11
1100 PRINT "A=";A;"B=";B :PRINT
1200 INPUT "<1>A>B <2>A<B <3>A=B (1/2/3)";NO
1300 ON NO GOSUB *SUB1, *SUB2, *SUB3
1400 PRINT "OK!":END
1500 PRINT "YOU ARE WRONG!":GOTO 1200
1600 '
1700 *SUB1
1800 IF A>B THEN RETURN 1400 ELSE RETURN 1500   ' 真なら1400行に戻る
1900 *SUB2                                      ' 偽なら1500行に戻る.
2000 IF A<B THEN RETURN 1400 ELSE RETURN 1500
2100 *SUB3
2200 IF A=B THEN RETURN 1400 ELSE RETURN 1500

結果：A=-10 B=-11

    <1>A>B <2>A<B <3>A=B (1/2/3) ? 2
     YOU ARE WRONG!
    <1>A>B <2>A<B <3>A=B ( /2/3) ? 1
     OK !
```

関連命令

GOSUB	：サブルーチンを呼び出します。
ON　GOSUB	：式の値により、サブルーチンを呼び出します。
ON　COM (n) GOSUB	：通信回線からの割り込み処理ルーチンを定義します。
ON　INTERVAL　GOSUB	：インターバル割り込み処理ルーチンを定義します。
ON　TIME　GOSUB	：タイマ割り込み処理ルーチンを定義します。
ON　MOUSE (n) GOSUB	：マウス割り込み処理ルーチンを定義します。
ON　KEY (n) GOSUB	：PFキー割り込み処理ルーチンを定義します。

RIGHT$関数

ライト・ダラー関数　*right$*

RIGHT$（文字式，バイト数）

機　能　文字列の右端から、指定されたバイト数分の文字列を取り出します。

説　明　＜文字式＞で指定した文字列の右端から、＜バイト数＞で指定した長さの文字列を取り出します。

＜文字式＞……取り出す文字列を指定します。

＜バイト数＞……取り出す文字列の長さを指定します。
指定する範囲は0～255です。

- ANK文字は１文字１バイト、日本語文字は１文字２バイトと数えます。
- ＜バイト数＞が＜文字式＞で示される文字列のバイト数より長い場合、取り出す文字列は＜文字式＞で指定した文字列になります。
- ＜バイト数＞に０を指定すると、取り出す文字列は空文字列になります。

例　題　指定した文字列の右端から７バイト分を取り出し表示します。

```
1000 A$="FUJITSU WORK STATION"
1100 B$=RIGHT$(A$,7)             ' 7 バイト分取り出す
1200 PRINT B$
1300 END

結果：STATION
```

関連命令　KRIGHT$関数：文字列の右端から、指定した文字数の文字列を取り出します。
LEFT$関数　：文字列の左端から、指定したバイト数の文字列を取り出します。
MID$関数　：文字列の中から指定したバイト数の文字列を取り出します。

RND

ランド　*random*　　計算

RND〔（式）〕

機　能　0以上1未満の乱数を返します。

説　明　乱数系列から乱数を取り出して返します。

<式>……乱数系列を正、負の数値または0で指定します。
省略すると、正の値を指定したとみなされます。

●<式>の値により，発生する乱数は次のようになります。

・<式>の値が負のとき……その負数に固有の乱数系列を作ります。この系列はRANDOMIZE命令によって変更するこはできません。したがって、プログラム実行ごとに別の系列から乱数が生成されるようにするには、毎回<式>の値が変わるように指定します。

・<式>の値が0のとき……1つ前のRND命令で返した値を返します。

・<式>の値が正のとき……同じ乱数系列の次の乱数を返します。乱数の系列は, RANDOMIZE命令により変更することができます。

例　題　繰り返し乱数を発生させ、その値を表示します。

行番号1400の式の値が負値のため、別の乱数系列が生成されます。

行番号1000～1200と、1500～1700では、乱数系列が異なるため、異なる乱数値が返されます。

RND（0）は、1つ前のRND命令で発生した乱数、すなわち行番号1600で最後に発生した値を返します。

```
1000 FOR I=1 TO 3
1100   PRINT RND;
1200 NEXT
1300 PRINT
1400 X=RND(-8)
1500 FOR I=1 TO 3
1600   PRINT RND;
1700 NEXT
1800 PRINT
1900 PRINT RND(0)
2000 END

結果： .297267  .449179  .269817
      .695043  .458227  2.85876E-02
      2.85876E-02
```

関連命令　RANDOMIZE：乱数の系列を変更します。

ROLL

ロール　*roll*　　　　画面表示

ROLL　〔上方向ドット数〕〔,〔左方向ドット数〕〔,{0｜1}〕〕

機　能　VRAMの内容を、上下および左右方向にスクロールします。

説　明　＜上方向ドット数＞または＜左方向ドット数＞で指定したドット数だけ、VRAM全体の内容を上下左右にスクロールします。

＜上方向ドット数＞……VRAMの縦方向のスクロール量をドット数で指定します。値が正のとき上方向（VRAMのデータが下方向）に、負のときに下方向（VRAMのデータが上方向）にスクロールします。
省略すると、上下方向へのスクロールを行いません。

＜左方向ドット数＞……VRAMの横方向のスクロール量をドット数で指定します。値が正のとき左方向（VRAMのデータが右方向）に、負のときに右方向（VRAMのデータが左方向）にスクロールします。
省略すると、左右方向へのスクロールを行いません。

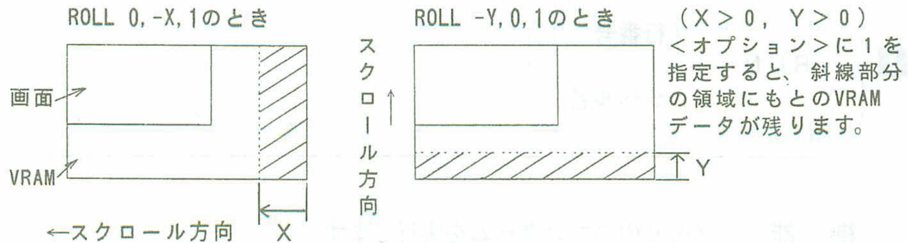
＜0＞、＜1＞　……スクロールによりVRAMのデータが移動した後に新たにVRAMに現れる領域を、背景色で塗りつぶすかどうかを指定します。

0：現在の背景色で塗りつぶします。

1：もとのVRAMのデータがそのまま残ります。

省略すると、0を指定したとみなされます。

●ROLL命令実行後のVRAM領域は以下のようになっています。



例　題　画面が下方向にスクロールするにつれ、正方形（データ）が上方向に移動します。

```
1000  CLS
1100  LINE (200,250)-(300,350),PSET,,BF
1200  FOR J=1 TO 10
1300    ROLL  -J,0,0             ' 上方向へのスクロールを行う
1400  NEXT J
1500  END
```

R

RUN〔省略形R.〕

ラン *run* コマンド・プログラムの分岐

1 RUN { 行番号 / ラベル名 }

機　能　メモリ内のプログラムを実行します。

説　明　オープン状態にある全てのファイルをクローズし、全ての数値変数を0に、文字変数を空文字列に初期化した後、プログラムを実行します。

<行番号> <ラベル名> ……実行を開始するプログラム行を指定します。
省略すると、プログラムの先頭から実行を開始します。

●プログラムの実行が終了する条件は、以下のとおりです。
・プログラムを最終行まで実行したとき。
・END命令かSTOP命令を実行したとき。
・BREAKキーまたはCTRL+Cキーを押したとき。

コンパイラ　●コンパイラでは、この形式のRUN命令は使用できません。

例　題　プログラムを実行します。

```
RUN                        ' プログラムを最初の行から実行する
RUN 50
                           ' プログラムを行番号50から実行する
```

2 RUN ファイルディスクリプタ〔,R〕

機　能　プログラムファイルから、プログラムを読み込んで実行します。

説　明　<ファイルディスクリプタ>……読み込むプログラムが格納されているファイルを指定します。（☞「●ファイルディスクリプタの形式」p. 31参照）

<R>……オープンされているファイルのクローズ処理を行わないで、プログラムを読み込んで実行する場合に指定します。省略すると、オープンされているファイルをクローズしてから、プログラムを実行します。

- プログラムの実行開始時には、自動的に全ての数値変数を0に、文字変数を空文字列に初期化します。
- プログラムの実行が終了する条件は、以下のとおりです。
  - ・プログラムを最終行まで実行したとき。
  - ・END命令かSTOP命令を実行したとき。
  - ・BREAKキーまたはCTRL+Cキーを押したとき。
- <R>を指定した場合は、ファイルを再度オープンすることなしに、割り付けられているファイル番号で、ファイルの入出力処理を行うことができます。

インタプリタ
- インタプリタでは、指定するプログラムはソースプログラムでなければなりません。

コンパイラ
- コンパイラでは、<R>は指定できません。
- コンパイラでは、指定するプログラムは実行形式プログラムでなければなりません。

例　題　プログラムを実行します。

```
RUN "PROG.BAS"                    ' メモリ上のプログラム"PROG.BAS"をロー
                                    ドして実行する
```

R

関連命令

CHAIN：指定したファイルのプログラムを呼び出します。そのとき、変数の引き渡しが可能です。

STOP ：プログラムの実行を中断し、エディタに戻ります。

END ：プログラムの実行を終了してファイルをクローズし、エディタに戻ります。

SAVE

セーブ *save*　　コマンド

```
SAVE　ファイルディスクリプタ〔, A〕
```

機　能　メモリ内のプログラムをファイルに保存します。

説　明　<ファイルディスクリプタ>で指定したファイルに、メモリ内のプログラムを保存します。

<ファイルディスクリプタ>……プログラムを保存するファイルを指定します。

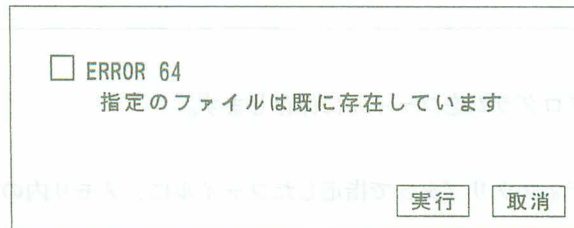
<A>……プログラムをアスキー形式（文字形式）で保存する場合に指定します。
省略すると、バイナリ形式（内部コード化された形式）で保存されます。

- バイナリ形式で保存すると、保存に要する時間が短くなり、ファイル容量も少なくてすみます。
- 保存したプログラムをMERGE命令で使用するためには、プログラムをアスキー形式で保存しておく必要があります。
- 処理終了後は、BASICエディタに戻ります。

コンパイラ
- コンパイラでは、SAVE命令は使用できません。SAVE命令を含むプログラムをコンパイルするとエラーになります。
- バイナリ形式で保存したプログラムはコンパイルできません。コンパイルするプログラムはアスキー形式で保存してください。

S

●＜ファイルディスクリプタ＞で指定したファイルが既に存在している場合は、確認のため次のメッセージが表示されます。



「実行」を選択すると、すでに存在しているファイルは削除され、メモリ内のプログラムが新しく保存されます。

「取消」を選択すると、すでに存在しているファイルをそのまま保存し、メモリ内のプログラムを保存しません。

関連命令　LOAD：プログラムをメモリに読み込みます。

SAVE@

セイブ・アットマーク　*save@*　　音楽／音声・画面表示

1

```
SAVE@ ファイルディスクリプタ
```

ただし、ファイル名の拡張子は.FMBです。

機　能　FM音源の音色データを、FM音源の専用メモリからファイルに格納します。

説　明　<ファイルディスクリプタ>……格納先のファイルを指定します。
拡張子が".FMB"のFM音源の音色データファイルを指定します。

●SAVE@命令は、FM音色（@1～@128）を全てファイルに格納します。

S

2

SAVE@　ファイルディスクリプタ，（開始位置）－（終了位置）
〔,〔パレット情報〕〔, 圧縮情報 〔,〔YCbCr〕〔,〔Y〕〔, C〕〕〕〕〕

ただし、ファイル名の拡張子は .TIFまたは .JPGです。

機　能　グラフィック画面の内容をイメージファイルに格納します。

説　明　<ファイルディスクリプタ>……格納先のファイルを指定します。
拡張子が".TIF"または".JPG"のファイルを指定します。

<（開始位置）－（終了位置）>……格納したい範囲を、オリジナルスクリーン座標で指定します。

<パレット情報>……グラフィックデータとともに、パレット情報を格納するかどうかを指定します。

0：グラフィックデータだけを格納し、パレット情報は格納しません。

1：グラフィックデータとともに、パレット情報を格納します。

省略すると、0を指定したとみなされます。

<圧縮情報>…………グラフィックデータを圧縮して格納するかどうかを指定します。

0：圧縮しないで格納します。

1：圧縮して格納します。

2：JPEGデータとして保存します。

省略すると、0を指定したとみなされます。

<YCbCr> ……輝度成分に対する色成分の圧縮比率を次の3つの中から選びます。

	輝度	色
1：	1：	1
2：	1：	1／2
4：	1：	1／4

省略すると、2を指定したとみなされます。

<Y>……輝度成分圧縮パラメータを指定します。
指定できる範囲は、0～100の数値です。
省略すると、25を指定したとみなされます。

<C>……色成分圧縮パラメータを指定します。
指定できる範囲は、0～100の数値です。
省略すると、25を指定したとみなされます。

- 32768色モードの場合には、パレット情報を指定できません。
- JPEGファイルを保存するときは、CLEAR命令でDLL領域を確保して下さい。
- V1.1のBASICでは、圧縮情報を指定できません。

関連命令　LOAD@：音色データファイルまたはイメージファイルを読み込みます。



32K色モードのときは、パレットフラグを指定した場合、エラーになります。
JPEGデータは画面モードが32K色モードの時以外扱えません。また、拡張子が.JPGでない場合は、エラーになります。
YCbCr、Y、Cの各パラメータは、圧縮情報を2に設定した時だけ設定します。それ以外の時に設定すると、エラーになります。
YCbCr、Y、Cの各パラメータは、それぞれ大きいほど圧縮率が高くなり、ファイルサイズは小さくなりますが、画質は劣化します。

S

3 SAVE@　ファイルディルクリプタ，配列名

機　能　配列の内容をファイルに格納します。

説　明　<ファイルディスクリプタ>……格納先のファイルを指定します。

<配列名>……格納したい配列を指定します。

SCREEN

スクリーン　*screen*　　　　画面表示

1

```
SCREEN 0
```

機　能　グラフィック２画面モードを解除し、テキスト画面を使用可能にします。

説　明　グラフィック２画面のうち、ページ１をテキスト画面に切り替えます。

- グラフィック２画面のどちらが前方にあっても、SCREEN 0 命令実行後は、テキスト画面が前側になります。
- マウスが使用されていれば、MOUSE 0 命令を実行したのと同様に、マウス機能が初期化されます。

2

```
SCRREN 1〔,〔アクティブページ〕〔,〔ディスプレイページ〕
                                〔,プライオリティ〕〕〕
```

機　能　グラフィック２画面モードに切り替えます。

説　明　テキスト画面をグラフィック画面（ページ１）に切り替えます。切り替えた画面は、16色モードです。

<アクティブページ>　……グラフィック画面のどのページに対してグラフィック命令を有効にするかを指定します。
0：ページ0に対して有効。
1：ページ1に対して有効。
省略すると、現在の設定値が保持されます。
BASIC起動時は、0に設定されています。

S

<ディスプレイページ>……グラフィック画面のどのページを表示するかを指定します。

0：どのページも表示しません。

1：ページ0のみを表示します。

2：ページ1のみを表示します。

3：両方のページを表示します。

省略すると、現在の設定値が保持されます。

BASIC起動時は、3に設定されています。

<プライオリティ>……グラフィック画面のどのページを前側にするかを指定します。

0：ページ0が前に出ます。

1：ページ1が前に出ます。

省略すると、現在の設定値が保持されます。

BASIC起動時は、1に設定されています。

- 256色モードでこの命令を使用すると、エラーになります。
- グラフィック2画面モードでは、SPRITE ON/OFFは、SPRITE OFFになります。
- グラフィック2画面モードではテキスト画面が使用できないので、SPRITE ON命令を実行したときと同じように、PRINT命令やINPUT命令のような、テキスト画面に文字を表示する命令は使用できません。
- ウィンドウやビューポートは、2画面のそれぞれで、独立に設定できます。
- マウスカーソルはMOUSE 0で指定された画面に表示されます。
- グラフィック2画面モードで、デジタイズ（SINPUT命令）ができるのは、次の場合です。
  - 32768色モード画面がアクティブページであること。
  - 16色モード画面が表示されていないこと。
- グラフィック2画面モードで、スーパーインポーズモード（SIMPOSE命令）にできるのは、16色モード画面が表示されていない場合です。
- V1.1のBASICでは、SCREEN命令は使用できません。

関連命令　SCREEN＠：画面モードを変更します。

SCREEN@

スクリーン・アットマーク　*screen@*　　画面表示

```
SCREEN@  画面モード [ , { 0 | 1 } [, (x , y )] ]
```

機　能　画面モードを変更します。

説　明　<画面モード>……変更モードを0～2で指定します。

0 ：16色モード
（画面ドット数 640×480 ドット）

1 ：32768色モード
（画面ドット数 320×240 ドット）

2 ：256色モード
（画面ドット数 640×480 ドット）

<0>、<1>……画面モードが1のときに、縮小表示するかどうかを指定します。

0 ：全画面表示をします。

1 ：縮小表示をします。VRAM全域の512×256ドットを1/4に縮小して表示します。

省略すると、0を指定したとみなされます。

< x , y >…………縮小表示を指定したときに、縮小表示する画面の左上隅の位置を、スクリーン座標値で指定します。指定できる範囲は次のとおりです。

x： 0～128
y： 0～224

省略すると、(0, 0) を指定したとみなされます。

●BASIC起動時には、SCREEN@ 0に設定されています。

●SCREEN@命令実行によって、以下のように設定が変更されます。

・画面は消去されます。

・PALETTE、ウィンドウ、ビューポートは初期状態になります。

・DEF PENは、DEF PEN 0, 1になります。

・SIMPOSE ON／OFFは、SIMPOSE OFFになります。

S

・SPRITE ON／OFFは、SPRITE OFFになります。
ただし、スプライトのキャラクタ番号、パターン、または色テーブルは保存されています。

- マウスが使用されていれば、MOUSE 0命令と同様に、マウス機能が初期化されます。
- 縮小表示は、640×480ドットの座標空間のうちの任意の位置に行うことができます。ただし、縮小表示された領域が、グラフィック画面の640×480ドット内に収まるように指定しなければなりません。
- 縮小表示を行っているときに、VRAM全域の512×256ドットに使っていない領域があると、その部分は空間になります。
- 縮小表示を行っているときに、スプライトを表示すると、一時的に全画面表示になりますが、表示後縮小表示に戻ります。

関連命令　SCREEN：グラフィック２画面モードに切り替えます。

SCREEN関数

スクリーン関数 *screen*

画面表示

```
SCREEN(x, y〔, セレクトスイッチ〕)
```

機　能　表示されている文字のキャラクタコードまたは属性を返します。

説　明　<x> <y>　……読み取る文字の位置をキャラクタ座標で指定します。

x：0～79

y：0～24

<セレクトスイッチ>……指定された文字のキャラクタコードを返すか、または属性についての情報を返すかを、0または1で指定します。

0：指定座標のキャラクタコードを返します。

1：指定座標の属性を返します。

省略すると、0を指定したとみなされます。

- 属性の値を8で割った余りが文字色を示します。8で割った商が0のときはノーマル表示、1のときはリバース表示を示します。
- 指定した座標に文字の表示がないときは、0を返します。
- 指定した値が画面の範囲外のときは、0を返します。

S

SEARCH

サーチ　*search*　　　　　　　　　　　　　　　　　　　　一般命令

```
SEARCH(配列変数名, 検索データ〔,〔開始要素番号〕〔,ステップ数〕〕)
```

機　能　配列変数の要素の中から＜検索データ＞を探し出し、その要素の要素番号（添字）を返します。

説　明

＜配列変数名＞　……DIM命令で宣言した配列を指定します。指定できる配列は一次元の配列だけです。

・文字型の場合

＜検索データ＞も文字型でなければなりません。

・文字型以外の場合

＜検索データ＞と配列の型が異なる場合は、＜検索データ＞の型を配列の型に変換してから検索します。

＜検索データ＞　……探し出す値を指定します。

＜開始要素番号＞……検索を開始する要素を指定します。
省略すると、0を指定したとみなされ、配列の先頭から検索されます。

＜ステップ数＞　……配列内の要素を、いくつおきに検索するかを正の整数で指定します。
省略すると、1を指定したとみなされ、＜開始要素番号＞以後にあるすべての要素が検索されます。

●指定した値が見つからない場合は、−1を返します。

例　題　配列要素の先頭から、指定した値を検索します。

```
1000  DIM A(20)
1100  FOR I=1 TO 20
1200    A(I)=I*10
1300  NEXT I
1400  B=50
1500  C=SEARCH(A,B)
1600  PRINT C
1700  END

結果： 5
```

関連命令　INSTR関数　：指定された文字列を検索し、先頭文字までのバイト数を返します。
KINSTR関数：指定された文字列を検索し、先頭文字までの文字数を返します。
DIM関数　　：配列変数を宣言し、メモリを割り当てます。

S

SGN

サイン *sign*　　　　計算

```
SGN (式)
```

機　能　＜式＞の結果の符号を−1、0、または1で返します。

説　明　＜式＞……数値式を指定します。

●返される値は以下のようになります。

−1 ：＜式＞が負のとき
 0 ：＜式＞が0のとき
 1 ：＜式＞が正のとき

例　題　入力した数値を判別し、0、1、−1のいずれかを表示します。

```
1000 INPUT A
1100 B=SGN(A)
1200 PRINT B
1300 END

結果：? 9
      1
      ? -40
      -1
      ? 0
      0
```

SHELL

シエル　*shell*

機械語プログラム

```
        ┌文字定数┐
SHELL  ┤        ├
        └文字変数┘
```

機　能　子プロセスの起動を行います。MS-DOSのコマンドの一部を起動することができます。

説　明　＜文字定数＞……起動したい子プロセスの実行形式やパラメタを文字定数で指定します。

＜文字変数＞……起動したい子プロセスの実行形式やパラメタを文字変数で指定します。

●現在、子プロセスとして起動できる実行形式（内部実行形式）には、以下のものがあります。

・指定されたサブディレクトリを作成する。
｛MKDIR｜MD｝　パス名

・指定されたサブディレクトリを削除する。
｛RMDIR｜RD｝　パス名

・指定されたディレクトリに移行する。
｛CHDIR｜CD｝　パス名

・カレントドライブを変更する。
ドライブ名：

例　題　例1：SHELL"MKDIR　B：¥SOURCE"

例2：CMDLINE$="MKDIR　B：¥SOURCE"
　　SHELL CMDLINE$

例3：SHELL "a："

S

SIMPOSE

エス・インポーズ　*super impose*

画面表示

1

```
SIMPOSE { ON  〔輝度〕 }
        { OFF        }
```

機　能　スーパーインポーズモードにするかしないかを設定します。

説　明　スーパーインポーズモードでは、グラフィック画面のスーパーインポーズビットが1の点(〔G, R, B, I〕の指定でI=1の点）に、ビデオ画面の画像が表示されます。

<ON>　……32768色モードのとき、スーパーインポーズモードにします。
ビデオ画面が表示され、テキスト画面が消えます。

<OFF>　……スーパーインポーズモードを解除します。
ビデオ画面が消え、テキスト画面が表示されます。

<輝度>　……ビデオ画面の明るさを0または1で指定します。
0：高輝度
1：低輝度
省略すると、0を指定したとみなされます。

- グラフィック2画面モードの場合は、2画面共16色モードであっても非表示にすれば、スーパーインポーズモードにすることができます。
- テキスト画面を使用するPRINT命令やINPUT命令を、スーパーインポーズモードで実行すると、エラーになります。
- SIMPOSE ONはパソコン本体にビデオカードが装着されていないと、正常に実行されません。
- 32768色モードの縮小表示は全画面表示に変わります。

例　題　ビデオ画面を表示します。

```
100 SCREEN@ 1
200 CLS
300 SIMPOSE ON 1
400 LINE(50,50)-(269,189),PSET,〔0,0,0,1〕,BF
500 GOTO 500
600 END
```

2 SIMPOSE 〔モード〕

機　能　ビデオ画像の表示モードを制御します。

説　明　スーパーインポーズモードまたはディジタイズモードに設定することができます。

＜モード＞……0～3の数値で、次のようなモードに設定されます。

0　：ビデオ画面を表示しません。

1または2：スーパーインポーズモード。
グラフィック画面のスーパーインポーズビットが1の点（〔G, R, B, I〕の指定でI=1の点）に、ビデオ画面の画像を表示します。

3　：ディジタイズモード
テレビまたはビデオの画像をディジタイズしてVRAMに取り込み、グラフィック画面に連続表示します。この状態で、＜モード＞に0～2を指定したSIMPOSE命令を実行すると、その時点の画像がVRAMに残ります。

省略すると、モードは変わりません。

- BASIC起動時は、SIMPOSE 0になっています。
- モード1～3は、パソコン本体にビデオカードが装着されていないと、正常に実行されません。
- SIMPOSE 3に類似の命令として、SINPUTがあります。SIMPOSE 3は、画面モードを移行する命令、SINPUTは、ビデオ画像をVRAMを取り込む実行文です。
- 32768色モードの縮小表示は全画面表示に変わります。

関連命令　SINPUT　：テレビまたはビデオの画像を、連続でVRAMに取り込み表示します。
SCREEN＠：画面モードを変更します。

SIN関数

サイン関数　*sine*　　　　計算

SIN（式）

機　能　三角関数サイン（正弦）の値を返します。

説　明　<式>……角度（単位：ラジアン）を指定します。

ラジアン＝π×角度（°）／180（°）

●式の型が倍精度の場合は倍精度の、単精度の場合は単精度の結果を返します。

例　題　入力した倍精度型の数値の、三角関数サインの値を求めて表示します。

```
1000 INPUT A
1100 B=SIN(3.14159/180*A)          ' 単精度の値をB に与える
1200 PRINT B
1300 INPUT A#
1400 B#=SIN(3.14159265359#/180*A#)  ' 倍精度の値をB#に与える
1500 PRINT B#
1600 END

結果：? 28.23
　　 .473012
　　 ? 59.62057485645
　　 .8626953298046
```

関連命令　COS関数 ：三角関数コサイン（余弦）の値を返します。
TAN関数：三角関数タンジェント（正弦）の値を返します。

SINPUT

エス・インプット　*super input*　　画面表示

```
SINPUT
```

機　能　テレビまたはビデオの画像を、連続でグラフィック画面に読み込み表示します。

説　明
- SINPUT命令を実行すると、テレビまたはビデオの画像がVRAMデータとなり、グラフィック画面に連続表示されます。
- マウスのボタンを押すか、キーボードのどれかのキーを押すと、SINPUT命令は実行を終了し、その時点の画像が静止画像としてグラフィック画面に残ります。



SINPUT命令は、32768色モード（SCREEN@ 1）のときだけ有効です。グラフィック2画面モードの場合は、32768色モード画面がアクティブページであることと、16色モード画面は非表示に設定されていることが必要です。
SINPUT命令は、ビデオカードが接続されていないときも、マウスのボタンを押すか、キーボードのどれかを押さないと終了しません。

例　題　テレビの画面データをファイルに保存します。

```
1000 SCREEN@ 1:CLS
1100 SINPUT       '   マウスボタンを押すと，次の文に進む．
1200 DIM DT%(319)
1300 OPEN "O",#1,"TV-DATA"
1400 FOR Y=0 TO 239
1500   GET@A (0,Y)-(319,Y),DT%
1600   FOR X=0 TO 319
1700     PRINT #1,MKI$(DT%(X));
1800   NEXT X
1900 NEXT Y
2000 CLOSE #1
2100 END
```

関連命令　SCREEN@：画面モードを変更します。

S

SMSGPLAY

サウンドメッセージプレイ　*sound message play*　音楽／音声

```
SMSGPLAY サウンドメッセージID
```

機　能　サウンドメッセージを設定します。

説　明　<サウンドメッセージID> ………OS側で登録されているサウンドメッセージID番号を指定します。

0：クリック
1：有効
2：無効
3：警告
4：注目
5：チャイム
6：エラー
7：効果

- サウンドメッセージは、TownsOSで供給されるシステムで、OS側でサウンドメッセージの使用が登録されている場合だけ、使用できます。
- インタプリンタの場合は、エラーのアラート表示にサウンドメッセージが再生されますが、エラー発生時にOS側でサウンドメッセージドライバが登録されていない場合、および登録されていてもOS側でサウンドメッセージOFFの設定がされている場合には、従来通りBEEPが鳴ります。
- コンパイラのエラー発生時にはサウンドメッセージは再生されません。
- V1.1のBASICでは、サウンドメッセージは再生されません。

S

SPACE$

スペース・ダラー　*space$*


```
SPACE$(個数)
```

機　能　空白文字列を返します。

説　明　＜個数＞で指定した数の空白文字列を返します。

＜個数＞……空白の数を0～255の数値で指定します。

● ＜個数＞に０を指定すると、空文字列を返します。

例　題　SPACE$で指定した長さの空白をあけて、表示します。

```
1000 A$="FUJITSU"+SPACE$(8)+"WORK STATION"
1100 PRINT A$
1200 END

結果：FUJITSU        WORK STATION
```

関連命令　STRING$関数：指定した１文字を並べた文字列を返します。
SPC関数　　：指定した数の空白を出力します。

SPC関数

エス・ピー・シー関数　*space*

```
SPC(個数)
```

機　能　指定した数の空白を出力します。

説　明　<個数>……空白の数を指定します。
指定できる範囲は0～255です。

●SPC関数はPRINT、LPRINT、WRITE、PRINT#およびWRITE#命令でのみ、使用できます。

●<個数>に0を指定すると空白をあけずに出力します。

例　題　文字列間に指定した個数の空白を入れて表示します。

```
1000  A$="富士通":B$="WORK":C$="STATION"
1100  PRINT A$;SPC(3);B$;SPC(2);C$
1200  END

結果：富士通   WORK  STATION
```

関連命令　TAB：指定した桁位置まで空白を出力します。

SPRITE

スプライト　*sprite*　　　　画面表示

DEF SPRITE命令で定義されたスプライトキャラクタを操作します。

1

```
SPRITE 0, キャラクタ番号, 表示
```

機　能　指定したスプライトキャラクタを表示または消去します。

説　明　<キャラクタ番号>……表示、または消去の対象にするスプライトキャラクタを指定します。

<表示>……対象となるスプライトキャラクタを、消去するか表示するかを０または１の数値で指定します。

０：消去する。
１：表示する。

●画面モードが16色モードか32768色モードかにかかわらず、16色モードのスプライトパターンおよび32768色モードのスプライトパターンのいずれでも表示することができます。



SPRITE ON後の初期状態は、<表示>に０が設定されています。したがって、スプライトパターンやスプライトキャラクタを定義しても、SPRITE命令（形式1）で<表示>に１を設定しないと、スプライトキャラクタは表示されません。

S

2 SPRITE　1，キャラクタ番号，パターン番号

機　能　指定したスプライトキャラクタを構成しているスプライトパターンを変更します。

説　明　＜キャラクタ番号＞……パターン変更の対象になるスプライトキャラクタを指定します。

＜パターン番号＞……スプライトキャラクタとして、新たに定義するスプライトパターンを指定します。
複数のスプライトパターンから成るスプライトキャラクタの場合は、先頭のパターン番号を指定します。

●スプライトパターンは、あらかじめDEF SPRITE 0命令で定義しておく必要があります。

3 SPRITE　2，キャラクタ番号，色テーブル番号

機　能　指定したスプライトキャラクタに割り当てられている色テーブル番号を変更します。

説　明　＜キャラクタ番号＞……色テーブル番号を変更するスプライトキャラクタを指定します。

＜色テーブル番号＞……指定したスプライトキャラクタに、新たに割り当てる色テーブルの番号を、0～255の範囲で指定します。

●色テーブルは、あらかじめDEF SPRITE 2命令またはDEF SPRITE 3命令で定義しておく必要があります。

4 SPRITE　3，キャラクタ番号，角度

機　能　指定したスプライトキャラクタを回転、または左右反転させます。

説　明　＜キャラクタ番号＞……回転、または左右反転の対象にするスプライトキャラクタを指定します。

＜角度＞……回転させる角度および左右反転させるかどうかを、0〜7の数値で指定します。

指定数値	回転角度	左右反転の有無
0	0 度	なし
1	180 度	あり
2	0 度	あり
3	180 度	なし
4	270 度	あり
5	270 度	なし
6	90 度	なし
7	90 度	あり

●複数のスプライトパターンで構成されるスプライトキャラクタでは、個々のスプライトパターンがそれぞれ回転するので、キャラクタ全体の回転にはなりません。

5 SPRITE　4，キャラクタ番号，縮小

機　能　指定したスプライトキャラクタを縮小します。

説　明　<キャラクタ番号>……縮小するスプライトキャラクタを指定します。

<縮小>　……縮小する方向と割合を、0～3の数値で指定します。

指定数値	垂直方向縮小	水平方向縮小
0	1倍	1倍
1	1倍	1／2倍
2	1／2倍	1倍
3	1／2倍	1／2倍

6 SPRITE　5，キャラクタ番号，スーパーインポーズ

機　能　指定したスプライトキャラクタをスーパーインポーズ表示するかどうかを指定します。

説　明　<キャラクタ番号>……スーパーインポーズ表示を変更するスプライトキャラクタを指定します。

<スーパーインポーズ>……指定したスプライトキャラクタをスーパーインポーズ表示するかどうかを、0または1で指定します。

0：通常の表示を行います。

1：スーパーインポーズ表示を行います。

> **用語**　スーパーインポーズ表示
> 透明と不透明を反転してスプライトパターンを透明にし、後のグラフィック画面が見えるようにする機能です。
> 32768色モードのスプライトでは、スプライトパターンの各ドットを定義する2バイトの中の最上位ビットが、そのドットを透明にするか不透明にするかを指定するビットです。また、16色モードのスプライトでは、間接色番号0（色テーブルの要素0）が透明色です。
> スーパーインポーズ表示を指定すると、透明色以外の部分が透明色になり、その部分にグラフィック画面が見えます。

●スーパーインポーズ指定の結果

①スプライト背景色が透明の場合（SPRITE ON命令で<色>を省略したとき）

スーパーインポーズ	0:通常表示	1:スーパーインポーズ表示
スプライトパターン内の不透明なドット	不透明（指定色のドットが見える。）	透明（*1）
スプライトパターン内の透明なドット	透明（*1）	
スプライトパターン外の透明なドット	透明（*1）	
効　果	グラフィック画面の上をスプライトパターンが見える。	グラフィック画面しか見えない（スプライト画面を表示する意味がない。）

*1　そのスプライトキャラクタより優先順位の低いスプライトキャラクタが、後ろに重なっていれば、そのスプライトキャラクタが見え、重なっていなければ、背景のグラフィック画面が見えます。

②スプライト背景色が不透明の場合（SPRITE ON命令で<色>を指定したとき）

スーパーインポーズ	1:スーパーインポーズ表示
スプライトパターン内の不透明なドット	透明（＊2）
スプライトパターン内の透明なドット	透明（＊1）
スプライトパターン外のドット	透明（＊1）
効　果	スプライトパターンの部分だけにグラフィック画面が見える（サーチライト風）

*1　そのスプライトキャラクタより優先順位の低いスプライトキャラクタが、後ろに重なっていれば、そのスプライトキャラクタが見え、重なっていなければ、スプライト背景色が見えます。

*2　背景のグラフィックが見えます。

7 SPRITE 6〔,〔キャラクタ番号〕〔,〔X〕〔,Y〕〕〕

機　能　指定したスプライトキャラクタを移動させます。

説　明　<キャラクタ番号>……移動させるスプライトキャラクタを指定します。
省略すると、オフセット参照キャラクタ(DEF SPRITE 1命令で<オフセット参照>の対象に設定したスプライトキャラクタ）が移動します。
（☞「●オフセット参照キャラクタの移動」p. 70参照）

<X>, <Y>　……スプライトキャラクタの移動量を、X方向、Y方向にドット数で指定します。
省略すると、0を指定したとみなされ、その方向には移動しません。

●スプライトキャラクタを画面の上端まで移動したとき、画面の上端 2 ドットの部分は表示されません。

画面モード	スプライト画面	表示できる範囲
16色モード	(0, 0) ～ (255, 255)	(0, 2) ～ (255, 255)
32768色モード	(0, 0) ～ (255, 239)	(0, 2) ～ (255, 239)



例　題　“●”をスプライトキャラクタとして定義し、色を変えながら移動します。

```
1000 '16color sprite
1100 SCREEN@ 0
1200 DIM A%(150),C%(16)
1300 SYMBOL (0,0),"☆",1,1,%1
1400 GET@A (0,0)-(15,15),A%
1500 DEF SPRITE 0,0,A%,0
1600 DEF SPRITE 1,0,(0,80),0,1,1,0,0
1700 DEF SPRITE 1,1,(240,150),0,1,1,0,31
1800 FOR I=0 TO 31:C%(1)=I*2^10+I*2^5+I:DEF SPRITE 2,I,C%:NEXT
1900 SCREEN@ 1
2000 SPRITE ON
2100 SPRITE 0,0,1        ' スプライトキャラクタを表示する
2200 SPRITE 0,1,1        ' スプライトキャラクタを消去する
2300 FOR X=0 TO 240
2400   SPRITE 2,0,X/8
2500   SPRITE 2,1,31-(X/8)
2600   FOR I=0 TO 50:NEXT
2700   SPRITE  6,0,1,0
2800   SPRITE  6,1,-1,0
2900 NEXT
```

関連命令

DEF　SPRITE：スプライトの各種情報を定義します。

SPRITE　ON／OFF：テキスト画面とスプライト画面の切り替えを行います。

SPRITE　SCREEN：グラフィック画面上のどの位置にスプライト画面を表示するかを指定します。

SPRITE関数：指定したスプライトキャラクタの情報を返します。

SPRITE　TIME：スプライト表示のタイミングを取り、複数パターンからなるスプライトキャラクタの移動時に、パターンが乱れないようにします。

SPRITE関数

スプライト関数　*sprite*

画面表示

1

```
SPRITE（キャラクタ番号, 0）
```

機　能　指定したスプライトキャラクタが表示されているか、消去されているかの情報を返します。

説　明　<キャラクタ番号>……対象になるスプライトキャラクタを指定します。

●返される値は以下のようになります。

0：指定したスプライトキャラクタは消去されています。

1：指定したスプライトキャラクタは表示されています。

2

```
SPRITE（キャラクタ番号, 1）
```

機　能　指定したスプライトキャラクタを構成するスプライトパターンの、先頭のパターン番号を返します。

説　明　<キャラクタ番号>……対象になるスプライトキャラクタを指定します。

S

3 SPRITE（キャラクタ番号, 2）

機　能　指定したスプライトキャラクタが使用している色テーブルの番号を返します。

説　明　<キャラクタ番号>……対象になるスプライトキャラクタを指定します。



この命令は、16色モードのスプライトキャラクタだけで有効です。

4 SPRITE（キャラクタ番号, 3）

機　能　指定したスプライトキャラクタの回転および左右反転状態を返します。

説　明　<キャラクタ番号>……対象になるスプライトキャラクタを指定します。

●返される値は以下のようになります。

返される値	回転角度	左右反転
0	0度	なし
1	180度	あり
2	0度	あり
3	180度	なし
4	270度	あり
5	270度	なし
6	90度	なし
7	90度	あり

5 SPRITE（キャラクタ番号，4）

機　能　指定したスプライトキャラクタの縮小状態を返します。

説　明　<キャラクタ番号>……対象になるスプライトキャラクタを指定します。
●返される値は以下のようになります。

返される値	垂直方向縮小	水平方向縮小
0	1倍	1倍
1	1倍	1／2倍
2	1／2倍	1倍
3	1／2倍	1／2倍

6 SPRITE（キャラクタ番号，5）

機　能　指定したスプライトキャラクタが、スーパーインポーズ表示になっているかどうかの情報を返します。

説　明　<キャラクタ番号>……対象になるスプライトキャラクタを指定します。

●返される値は以下のようになります。
0：通常の表示になっています。
1：スーパーインポーズ表示になっています。

S

7 SPRITE（キャラクタ番号, 6）

機　能　指定したスプライトキャラクタの現在の表示位置のX座標をスプライト座標値で返します。

説　明　<キャラクタ番号>……対象になるスプライトキャラクタを指定します。



指定したスプライトキャラクタが、オフセット参照キャラクタの場合は、現在の表示位置が返されないことがあります。
SPRITE関数（形式7）は、オフセット参照で移動した位置ではなく、最後にキャラクタ番号を指定して移動した位置のx座標値を返します。

8 SPRITE（キャラクタ番号, 7）

機　能　指定したスプライトキャラクタの現在の表示位置のY座標をスプライト座標値で返します。

説　明　<キャラクタ番号>……対象になるスプライトキャラクタを指定します。



指定したスプライトキャラクタが、オフセット参照キャラクタの場合は、現在の表示位置が返されないことがあります。
SPRITE関数（形式8）は、オフセット参照で移動した位置ではなく、最後にキャラクタ番号を指定して移動した位置のy座標値を返します。

9 SPRITE（キャラクタ番号, 8）

機　能　指定したスプライトキャラクタを構成するスプライトパターンのうち、横方向のスプライトパターンの数を返します。

説　明　<キャラクタ番号>……対象になるスプライトキャラクタを指定します。

10 SPRITE（キャラクタ番号, 9）

機　能　指定したスプライトキャラクタを構成するスプライトパターンのうち、縦方向のスプライトパターンの数を返します。

説　明　<キャラクタ番号>……対象になるスプライトキャラクタを指定します。

関連命令

DEF SPRITE	：スプライトの各種情報を定義します。
SPRITE ON／OFF	：テキスト画面とスプライト画面の切り替えを行います。
SPRITE SCREEN	：グラフィック画面上のどの位置にスプライト画面を表示するかを指定します。
SPRITE	：スプライトキャラクタを操作します。
SPRITE TIME	：スプライト表示のタイミングを取り、複数パターンからなるスプライトキャラクタの移動時に、パターンが乱れないようにします。

S

SPRITE ON/OFF

スプライト　オン/オフ　*sprite on/off*　　画面表示

```
SPRITE { ON〔色〕
       { OFF
```

機　能　テキスト画面とスプライト画面の切り替えを行います。

説　明　<色>……スーパーインポーズ表示のときのスプライト画面の背景色を、輝度（R、G、B）または色番号（0～7）で指定します。
省略すると、0を指定したとみなされ、画面の背景色を透明にします。

- SPRITE　ON　テキスト画面を消してスプライト画面を表示します。
- SPRITE　OFF　スプライト画面を消してテキスト画面を表示します。
- <色>にパレット番号（%n）を指定することはできません。
- SPRITE　ONを指定しても、この命令を含むプログラムが終了または中断したときは、自動的にSPRITE　OFFモードに切り替わります。



・<色>の指定は、スーパーインポーズ表示（SPRITE　5）を行う場合にだけ有効です。
・SPRITE　ONの状態では、PRINT命令やINPUT命令のような、テキスト画面に文字を表示する命令は使用できません。

関連命令

DEF　SPRITE	：スプライトの各種情報を定義します。
SPRITE　SCREEN	：グラフィック画面上のどの位置にスプライト画面を表示するかを指定します。
SPRITE	：スプライトキャラクタを操作します。
SPRITE関数	：指定したスプライトキャラクタの情報を返します。
SPRITE　TIME	：スプライト表示のタイミングを取り、複数パターンからなるスプライトキャラクタの移動時に、パターンが乱れないようにします。

SPRITE SCREEN

スプライト・スクリーン　*sprite screen*　　画面表示



```
SPRITE SCREEN (X, Y)
```

機　能　グラフィック画面上のどの位置にスプライト画面を表示するかを指定します。

説　明　<X><Y>　……スプライト画面の左上隅を、グラフィック画面のどこに重ねるかを、オリジナルスクリーン座標で指定します。

> 注意　スプライト画面の一部がグラフィック画面からはみだす座標を指定するとエラーになります。

- SPRITE SCREEN命令を実行しない場合には、グラフィック画面の左上隅（0, 0）を指定したとみなされます。
- 表示位置の指定範囲は、画面のモードにより異なります。

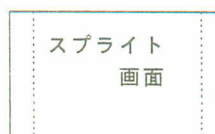
・32768色モードの場合

X：0〜64

Y：0

〔指定例〕





S

・16色モード

X：0～384

Y：0～224

〔指定例〕

SPRITE SCREEN (0, 0)

SPRITE SCREEN (280, 120)

2 SPRITE SCREEN {0 / 1}

機　能　32768色モード（SCREEN@ 1）におけるスプライト画面の大きさを変更します。

説　明　＜0＞＜1＞……スプライト画面を拡大して、グラフィック画面と同じ大きさにするかどうかを指定します。

0：拡大しない。
スプライト画面のドット縦横比は1対1で、画面の大きさはグラフィック画面より横方向が小さくなります。

1：拡大する。
スプライト画面のドット縦横比は4対5で、画面の大きさはグラフィック画面と同じになります。

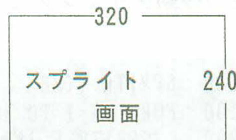
●SPRITE　SCREEN命令（形式2）は、32768色モードのときだけ有効です。
●SPRITE　ONを実行した直後は、0が設定されています。
●SPRITE　SCREEN命令（形式2）の指定例を図で説明します。

SPRITE SCREEN 0



グラフィック画面の4/5 にスプライト画面が表示されます.

SPRITE SCREEN 1



画面全体にスプライト画面が表示されます.

関連命令

DEF　SPRITE　：スプライトの各種情報を定義します。
SPRITE　ON／OFF：テキスト画面とスプライト画面の切り替えを行います。
SPRITE　：スプライトキャラクタを操作します。
SPRITE関数　：指定したスプライトキャラクタの情報を返します。
SPRITE　TIME　：スプライト表示のタイミングを取り、複数パターンからなるスプライトキャラクタの移動時に、パターンが乱れないようにします。

SPRITE TIME

スプライト　タイム　*sprite time*　　画面表示

```
SPRITE　TIME
```

機　能　スプライト表示のタイミングを取り、複数のスプライトパターンからなるスプライトキャラクタの移動時に、パターンが乱れないようにします。

説　明　複数のスプライトパターンからなるスプライトキャラクタを移動する直前に、この命令を実行し、パターンが乱れないようにします。



SPRITE TIME命令を実行後は、なるべく早くスプライトキャラクタの移動を行う必要があります。
また、一度の移動タイミングで全部のキャラクタが移動しきれないときはパターンが乱れることがあります。このような場合には、一度に移動するキャラクタの数を減らして、再度SPRITE TIME命令を実行し、タイミングを取り直してから、残りのキャラクタを移動するようにします。

例　題　一度の移動タイミングで10個のキャラクタを移動します。

```
1000  SPRITE TIME
1100  FOR I%=1 TO 10
1200    SPRITE 6,I%*20,3,0
1300  NEXT
```

関連命令

DEF　SPRITE　　：スプライトの各種情報を定義します。
SPRITE　ON／OFF：テキスト画面とグラフィック画面の切り替えを行います。
SPRITE　SCREEN　：グラフィック画面のどの位置にテキスト画面を表示するかを指定します。
SPRITE　　　　　：スプライトキャラクタを操作します。
SPRITE関数　　　：指定したスプライトキャラクタの情報を返します。

SQR

スクエア・ルート　*square root*

SQR(式)

機　能　平方根を返します。

説　明　<式>……0または正の実数を指定します。

●式の型が、倍精度の場合は倍精度の、単精度の場合は単精度の結果を返します。

例　題　入力した数値の平方根を求め表示します。

```
1000 INPUT A
1100 B=SQR(A)
1200 PRINT B
1300 INPUT A#
1400 B#=SQR(A#)
1500 PRINT B#
1600 END

結果：? 56
      7.48331
      ? 20.32156987
      4.50794519376622
```

STOP

ストップ　*stop*


```
STOP
```

機　能　プログラムの実行を中断します。

説　明　実行中のプログラムを中断します。

インタプリタ
- インタプリタでは、中断後、エディタのサブメニューから「Continue」を選択すると、中断された次の命令から実行を再開することができます。
- インタプリタでは、プログラムを中断する際、ファイルのクローズ処理は行われません。

コンパイラ
- コンパイラでは、中断したプログラムを再開することはできません。

関連命令　END：プログラムの実行を終了し、オープンされている全てのファイルをクローズ処理します。

STOP ON/OFF

ストップ・オン・オフ　*stop on/off*

プログラムの分岐

```
STOP {ON
      OFF}
```

機　能　BREAKキーまたは、INPUT命令によるデータ入力待ち状態でのCTRL+Cキーによる、プログラムの中断を有効にしたり、無効にしたりします。

説　明

- STOP ON　プログラムの中断を有効にします。
- STOP OFF　プログラムの中断を無効にします。
- プログラムの実行開始時は、STOP ON状態になっています。



STOP OFFの状態では、実行中のプログラムを中断する手段がありません。特に、デバッグ途中のプログラムなどの実行では、十分に注意してください。

S

例　題　STOP ONと、STOP OFFを交互に使用して処理を実行します。

```
1000 STOP OFF
1100 PRINT "STOP OFF"
1200 FOR I=1 TO 5000:NEXT  'この間は, BREAK キーを押しても停止しない
1300 STOP ON
1400 PRINT "STOP ON"
1500 FOR J=1 TO 5000:NEXT  'この間は, BREAK キーを押すと停止する
1600 GOTO 1000

結果：STOP OFF
      STOP ON
      STOP OFF
      STOP ON
      STOP OFF
      STOP ON
```

STR$関数

エス・ティー・アール・ダラー関数　*string$*

文字列処理

```
STR$(式)
```

機　能　数値を、数字の文字列に変換します。

説　明　<式>……数値式を指定します。

例　題　入力された数値を表す文字列を表示します。

```
1000 INPUT A
1100 B$="A="+STR$(A)  ' 文字定数"A="に入力された数値を連結する
1200 PRINT B$
1300 END

結果：? 4569
      A= 4569
```

関連命令

VAL関数	：STR$関数とは逆に、数字の文字列を数値に変換します。
MKI$／MKL$／MKS$／MKD$関数	：数値を文字列に変換します。
CVI／CVL／CVS／CVD関数	：文字列を数値に変換します。

STRING$

ストリング・ダラー　*string$*

STRING$（文字式, ｛キャラクタコード / 文字式｝）

機　能　指定した文字を並べた文字列を返します。

説　明　＜文字式＞または＜キャラクタコード＞で指定した1文字を＜文字数＞だけ並べた文字列を返します。

＜文字数＞　…… 文字列の長さを0～255の数値で指定します。

＜キャラクタコード＞……文字をキャラクタコードで指定します。

＜文字式＞　……1文字を指定します。
2文字以上の文字列を指定した場合は、先頭の1文字だけが有効です。

●＜文字数＞に0を指定すると、空文字列を返します。

例　題　指定した文字と長さで作られた文字列を表示します。

```
1000 A$=STRING$(20,"A")
1100 B$=STRING$(20,"AB")  ' "A" のみ有効となる
1200 PRINT A$
1300 PRINT B$
1400 END

結果：AAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAA
```

関連命令　SPACE$：空白文字列を返します。

SWAP

スワップ　*swap*　　　　文字列処理

SWAP　変数，変数

機　能　同じ型の２つの変数の値を交換します。

説　明　＜変数＞……整数、ロング型整数、単精度実数、倍精度実数、または文字の、いずれかの変数を指定します。また、配列変数の要素も指定することができます。

２つの＜変数＞の型は同じでなければなりません。

例　題　同じ型の変数A、Bの値を入れ換えて表示します。

```
1000 INPUT A, B
1100 PRINT "A="; A, "B=";B
1200 PRINT "------------------------"
1300 SWAP A, B
1400 PRINT "A="; A, "B=";B
1500 END
```

```
結果：? 100,800
      A= 100              B= 800
      ------------------------
      A= 800              B= 100
```

S

SYMBOL

シンボル *symbol*　　画面表示

```
SYMBOL {(wx, wy) | STEP (x, y)} , 文字列, 横倍率, 縦倍率〔, 〔色〕
〔, 〔角度コード〕〔, 〔論理操作〕〔, 〔文字修飾〕〔, 文字間隔〕〕〕〕〕
```

機　能　文字列をグラフィック画面に拡大または縮小表示します。また、角度を変えて表示します。

説　明　<wx, wy> ……<文字列>を表示する枠の左上隅を、ワールド座標値で指定します。

<STEP（x, y）> ……最終参照座標（LP）からの距離（x, y）で指定します。

<文字列> ……拡大、縮小表示する文字列を指定します。

<横倍率> ……表示する文字列の横の倍率を数値で指定します。
ANK文字は<横倍率>×８のドット数で、日本語文字は<横倍率>×16のドット数で表示されます。
倍率は255以下の実数で指定します。小数の指定もできます。

<縦倍率> ……表示する文字列の縦の倍率を数値で指定します。
ANK文字、日本語文字共に<縦倍率>×16のドット数で表示されます。
倍率は255以下の実数で指定します。小数の指定もできます。

<色> ……拡大、縮小表示する文字列の色を指定します。
省略すると、直前に実行したCOLOR命令で指定した<前景色>で表示します。

<角度コード>……<文字列>を回転する角度を、0～3の数値で指定します。

0：ノーマルに表示します。

1：90°左回転します。

2：180°左回転します。

3：270°左回転します。

省略すると、0を指定したとみなされます。

<論理操作>……PSET、PRESET、AND、OR、XOR、NOT、PASTEL、OPAQUEの、いずれかを指定します。

省略すると、PSETを指定したとみなされます。

<文字修飾>……表示する文字列の属性を、属性値の合計で指定します。1回の指定で、同じ属性値を2回以上加算できません。

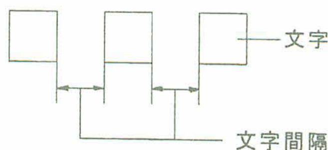
属性値は以下のとおりです。

0	通常	
1	太文字	(右に1ドット太くなります)
2	斜体	(右ななめに傾きます)
4	影付	(右ななめ下に影を付けます)
8	縁取り	(フチを付けます)
16	アンダーライン	(文字の最下部に線を引きます)
32	オーバーライン	(文字の最上部に線を引きます)
64	消し線	(文字の中心に線を引きます)

省略すると、0を指定したとみなされます。

<文字間隔>……文字と文字の間の長さをドット数で指定します。

省略すると、0ドットを指定したとみなされます。



S

●<文字修飾>で指定された影や縁取りは、背景色で描きます。したがって、文字の後ろに背景色以外の色がないと見えません。

●アンダーライン、オーバーライン、消し線は、指定した<色>で描きます。

●<文字修飾>の指定例を以下に示します。

例　SYMBOL（x, y）"123", …　,pset, 6
└2＋4　斜体にして影をつける

●SYMBOL命令で表示される文字のフォントを、DEF FONTで指定することができます。ただし、<角度コード>に0以外を指定した場合は、DEF FONTの指定にかかわらず、16ドットフォントで描画されます。

例　題　文字列を表示します。

```
1000  CLS
1100  SYMBOL (150,50), "富士通",4,2,5
1200  SYMBOL (100,50), "FUJITSU",2,2,3,3,PSET
1300  END
```

関連命令　DEF FONT：文字のフォントを指定します。

SYSTEM

システム　*system*　　　　プログラムの分岐

```
SYSTEM
```

機　能　BASICを終了します。

説　明　クローズされていないファイルをクローズして、BASICを終了し、TownsMENUに戻ります。

- BASICエディタで右上のEXITボタンをクリックした場合と結果は同じになります。

例　題　BASICを終了して、TownsMENUに戻ります。

```
SYSTEM
```

関連命令　END：プログラムを終了し、全てのファイルをクローズして、BASICエディタの画面に戻ります。

S

TAB

タブ *tab* 画面表示・印刷

TAB（桁位置）

機　能　現在の桁位置から指定した桁位置まで空白を出力します。

説　明　<桁位置>……どこまで空白にするかを指定します。
指定できる範囲は0～2147483647です。

- PRINT、LPRINT、WRITE、PRINT#、およびWRITE#命令でのみ使用できます。
- <桁位置>に0を指定すると、次行の先頭から出力します。
- <桁位置>が1行の表示文字数を超えた場合は、1行の文字数で割った余りを実際の桁位置にします。
- <桁位置>が現在の桁位置より小さい場合、または1行の文字数で割った余りが現在の桁位置より小さい場合は、次行の先頭から指定した桁位置までが空白になります。

例　題　文字列間に指定した桁位置まで空白を入れながら表示します。

```
1000  A$="富士通":B$="WORK":C$="STATION"
1100  PRINT A$;TAB(10);B$;TAB(25);C$
1200  END

結果：富士通    WORK           STATION
```

関連命令　SPC関数：指定した数の空白を出力します。

TAN関数

タンジェント関数　*tangent*

計算

TAN（式）

機　能　三角関数タンジェント（正接）の値を返します。

説　明　＜式＞……角度（単位：ラジアン）を指定します。

ラジアン＝π×角度（°）／180（°）

●式の型が、倍精度の場合は倍精度の、単精度の場合は単精度の結果を返します。

例　題　入力した倍精度型の数値の、三角関数タンジェント値を求め表示します。

```
1000 INPUT A
1100 B=TAN(3.14159!/180*A)              ' 単精度の値をB に与える
1200 PRINT B
1300 INPUT A#
1400 B#=TAN(3.14159265359#/180*A#)      ' 倍精度の値をB#に与える
1500 PRINT B#
1600 END

結果：? 50
      1.19175
      ? 23.36621489
      .432038737746373
```

関連命令　SIN関数　：三角関数サイン（正接）の値を返します。

COS関数　：三角関数コサイン（余弦）の値を返します。

ATN関数：三角関数アークタンジェント（逆正接）の値を返します。

TIME

タイム　*time*

時計

```
TIME　文字式
```

機　能　タイマ割り込みの時刻を設定します。

説　明　＜文字式＞……タイマ割り込みの時刻を、文字列″hh:mm:ss″で指定します。

hh　：時を00～23の数値で指定します。

mm：分を00～59の数値で指定します。

ss　：秒を00～59の数値で指定します。

- タイマ割り込みの時刻は1つだけ設定できます。タイマ割り込み時刻の設定後に、別のタイマ割り込みの時刻を設定すると、後に設定した時刻が有効になります。
- タイマ割り込みは、TIME ONのとき許可され、TIME OFF命令で禁止されます。
- タイマ割り込みルーチンの実行中は、TIME STOP状態になっています。

例　題　ON　TIME　GOSUB命令の例を参照してください。

関連命令　ON　TIME　GOSUB　:タイマ割り込み処理ルーチンを定義します。

TIME　ON／OFF／STOP :タイマ割り込みを許可、禁止または停止します。

TIME関数

タイム関数　*time*

時計

```
TIME
```

機　能　BASICシステムタイマが示す時刻を秒で返します。

説　明　BASICシステムタイマが示す時刻を、00:00:00時点（同日の午前0時）からの秒数で返します。

例　題　現在の時刻を、秒と時分秒の2種類で表示します。

```
1000  PRINT TIME           ' 時刻を秒で表示する.
1100  PRINT TIME$          ' 時刻を時分秒で表示する.
1200  END
結果：       417838

           11:35:38
```

関連命令　TIME$：BASICシステムタイマが示す時刻を返すか、または変更します。

T

TIME$

タイム・ダラー *time$*　　時計

1　TIME$

機　能　BASICシステムタイマが示す時刻を、時、分、秒で返します。

説　明　時刻をhh:mm:ssの形式の文字列で返します。

hh　：時（00～23）
mm：分（00～59）
ss　：秒（00～59）

例　題　BASICシステムタイマが示す時刻を表示します。

```
1000 PRINT TIME$
結果：12:15:00
```

2 TIME$＝文字式

機　能　BASICシステムタイマの時刻を変更します。

説　明　＜文字列＞……変更する時刻を文字列 "hh:mm:ss" で指定します。

hh　：時を00～23で指定します。

mm：分を00～59で指定します。

ss　：秒を00～59で指定します。



変更した時刻は、BASICの実行環境だけで有効で、いったんBASICを終了すると、変更前の値に戻ります。

例　題　BASICシステムタイマの時刻を表示し、その値を変更した後、新しく設定した時刻を表示します。

```
1000 PRINT TIME$                    ' 時刻を表示する
1100 TIME$="13:00:00"               ' 時刻を変更する
1200 PRINT TIME$                    ' 変更後の時刻を表示する
1300 END

結果：12:15:00
      13:00:00
```

関連命令　DATE$：BASICシステムタイマが示す日付を返すか、または変更します。

TIME ON/OFF/STOP

タイム・オン/オフ/ストップ　*time on/off/stop*　　プログラムの分岐・時計

```
TIME {ON
      OFF
      STOP}
```

機　能　タイマ割り込みを許可、禁止または停止します。

説　明

●TIME　ON

タイマ割り込みを許可します。割り込み時刻は、TIME命令で設定します。割り込みが発生すると、ON TIME GOSUB命令で定義するタイマ割り込み処理ルーチンが実行されます。

●TIME　OFF

タイマ割り込み動作を禁止します。

●TIME　STOP

タイマ割り込み動作を一時停止します。割り込みがかかっても、その時点では割り込み動作は行われず、次にTIME ON命令が実行されたときに割り込み動作が行われます。タイマ割り込み処理ルーチンの実行中はTIME STOP状態になっています。

例　題　ON　TIME　GOSUBを参照してください。

関連命令

ON　TIME　GOSUB	：タイマ割り込み処理ルーチンを定義します。
TIME	：タイマ割り込み時刻を設定します。
TIME$	：BASICシステムタイマが示す時刻を返すか、または変更します。

TRON/TROFF

トレース・オン/オフ　*trace on/trace off*

一般命令

```
┌ TRON  ┐
│       │
└ TROFF ┘
```

機　能　トレースモードにするか、またはトレースモードを削除します。

説　明
- TRONを実行すると、トレースモードになります。
  トレースモードになると、プログラム実行時に、プログラムの行番号（トレース行）を表示してからその行を実行します。
  トレースモードの解除は、次のようにします。
  - ・TROFFを実行する。
  - ・NEWを実行する。
  - ・エディタの「新規作成」を選択する。
  - ・エディタの「RUN」を選択する。
- TROFFを実行すると、トレースモードが解除になります。
- TROFF命令は、割り込み処理ルーチンおよびエラー処理ルーチンでは、使用できません。使用するとエラーになります。
- 初期設定はTROFFになっています。
- BASICエディタの「実行」の「RUN(TRON)」を指定すると、TRON命令に続いてRUN命令を実行したのと同じ結果になります。

インタプリタ
- インタプリタでは、プログラム中に記述されている全ての行番号を表示します。

コンパイラ
- コンパイラでは、ソースプログラム中の非実行文については行番号を表示しません。ただし、非実行文と実行文が同一行に存在するときは、行番号を表示します。

T

例　題　プログラムをトレースモードで実行します。

```
1000 INPUT A$
1100 PRINT A$
1200 END
TRON
RUN                    ＴＲＯＮの状態でメモリ上のプログラム
                       を実行する
結果：〔1000〕？富士通
      〔1100〕富士通
      〔1200〕
```

VAL関数

バリュー関数　*value*

文字列処理

```
VAL（文字式）
```

機　能　数字の文字列を数値に変換します。

説　明　<文字式>……数字の文字列を指定します。

- 文字列内の空白は読み飛ばします。
- 文字列内の最初の文字が、+、−、&、数字または空白でない場合は、0を返します。
- 文字列内に数字（16進数の場合は0〜9、A〜Fの文字、8進数の場合は0〜7）以外の文字が現れた場合は、それ以降の文字を無視します。

例　題　数字の文字列を数値に変換して、加算した値を表示します。

```
1000 A$="100":B$="50"
1100 A=VAL(A$):B=VAL(B$)        ' 文字変数A$, B$を数値に変換する
1200 C=A+B                      ' 変換した数値を加算する
1300 PRINT C
1400 END

結果：150
```

関連命令

STR$関数	：VAL関数とは逆に、数値を文字列に変換します。
CVI／CVL／CVS／CVD関数	：文字列を数値に変換します。
MKI$／MKL$／MKS$／MKD$関数	：数値を文字列に変換します。

VARPTR関数

バーポインタ関数　*variable pointer*


```
VARPTR（変数名）
```

機　能　変数のアドレスを返します。

説　明　<変数名>で指定した変数のアドレスをロング型整数で返します。

<変数名>……アドレスを取得したい変数を指定します。

●返される値の範囲は、0～4294967295です。

VIEW

ビュー *view*　　　　画面表示

```
VIEW〔(sx1, sy1) − (sx2, sy2)〔,〔領域色〕〔, 境界色〕〕〕
```

機　能　ビューポートの範囲を指定します。

説　明　ビューポートの左上隅と右上隅の2つの頂点を、オリジナルスクリーン座標値で指定します。

＜sx1, sy1＞ ＜sx2, sy2＞……ビューポートの左上隅、ビューポートの右下隅のオリジナルスクリーン座標を指定します。
指定できる範囲は画面モードにより異なります。
16色モードまたは256色モード:(0,0)～(1023,511)
32768色モード:(0,0)～(511,255)
省略すると、オリジナルスクリーン座標の(0,0)を左上隅とする初期設定の状態になります。

＜領域色＞　……ビューポート内を塗りつぶす色を指定します。
（☞「●色の指定」p. 54参照）
省略すると、塗りつぶしません。

＜境界色＞　……ビューポートの枠の色を指定します。
（☞「●色の指定」p. 54参照）
省略すると、枠は描かれません。

- BASIC起動時、またはSCREEN@命令実行直後のビューポート初期設定は、以下のようになっています。
  16色モードまたは256色モード：（0, 0）～（639, 479）
  32768色モード　　　　　　　：（0, 0）～（319, 239）
- ＜sx1＞が＜sx2＞より大きいとき、＜sy1＞が＜sy2＞より大きいとき、またはこれらの値がオリジナルスクリーン座標の範囲を超えている場合は、エラーになります。

●一度設定されたビューポートは、次にVIEW命令またはSCREEN@命令が実行されるまで有効です。

●VIEW命令の実行後、最終参照座標（LP）は、ビューポートの左隅上の頂点と対応しているワールド座標値に移動します。

●VIEW命令の実行後、DEF　PEN命令で指定したペンの太さが初期化され、DEF　PEN　0,1になります。

例　題　設定したビューポートの座標値を、左上隅のX座標、左上隅のY座標、右下隅のX座標、右下隅のY座標の順に表示します。

```
1000  VIEW (0,0)-(639,479)
1100  FOR I=0 TO 3
1200    S=VIEW(I)
1300    PRINT S
1400  NEXT
1500  END

結果： 0
       0
       639
       479
```

関連命令　WINDOW：ビューポートに表示するワールド座標内のウィンドウの範囲を指定します。

VIEW関数：現在ビューポートが設定されている位置を返します。

VIEW関数

ビュー関数　*view*

画面表示

VIEW(機能)

機　能　現在ビューポートが設定されている位置を返します。

説　明　VIEW命令で設定されている現在のビューポートの範囲を、オリジナルスクリーン座標値で返します。

＜機能＞……ビューポートの左上隅の座標値を（sx1, sy1）、右下隅を（sx2, sy2）として、この4つの座標の内のどれを返すかを、0～3の数値で指定します。

0：ビューポートの左上隅のＸ座標（sx1）を返す。
1：ビューポートの左上隅のＹ座標（sy1）を返す。
2：ビューポートの右下隅のＸ座標（sx2）を返す。
3：ビューポートの右下隅のＹ座標（sy2）を返す。

例　題　設定したビューポートの座標値を、左上隅のX座標、左上隅のY座標、右隅下のX座標、右隅下のY座標の順に、画面に表示します。

```
1000  VIEW (0,0)-(639,479)
1100  FOR I=0 TO 3
1200    S=VIEW(I)
1300    PRINT S
1400  NEXT
1500  END

結果： 0
       0
       639
       479
```

関連命令　VIEW　：ビューポートの範囲を指定します。
MAP関数：ワールド座標をスクリーン座標に、スクリーン座標をワールド座標に変換します。

V

VOICE

ボイス *voice*


```
VOICE  音色番号, 配列名 [, {0|1}]
```

機　能　音源の音色を設定します。

説　明　指定した配列の値を音色パラメータとして、FM音源またはPCM音源の＜音色番号＞の音色を設定します。配列の値が、FM音源またはPCM音源の専用メモリに読み込まれます。

＜音色番号＞……音色データを設定する音色番号を指定します。
指定範囲は、音色データがFM音源であるかPCM音源であるかによって、以下のように制限されます。

FM音源　：1～128
PCM音源：1～32

＜配列名＞　……音色データが読み込まれている文字型以外の配列を指定します。
配列の大きさは、FM音源であるかPCM音源であるかによって、以下のように制限されます。

FM音源　：・48バイト以上の配列を指定します。
・整数配列の場合、DIM 配列名 (23) で、大きさは48バイトです。

PCM音源：・128バイト以上の配列を指定します。
・整数配列の場合、DIM 配列名 (63) で、大きさは128バイトです。

＜０＞＜１＞……音源の種類を指定します。

０：FM　音色データのとき
１：PCM音色データのとき

省略すると、０を指定したとみなされます。

> **注意**
> PCM音色は、インスツルメントとサウンドデータで構成されていますが、VOICE命令で設定するのは、このうちインスツルメントの方だけです。

- VOICE命令の実行後、PLAY命令のMML@コマンドで＜音色番号＞を指定すると、＜配列名＞が示す音色で演奏します。

関連命令　VOICE COPY：音色データを配列変数に複写します。
PLAY　　　　：音楽の演奏を行います。

VOICE COPY

ボイス・コピー　*voice copy*　　　　音楽/音声

```
VOICE COPY 音色番号, 配列名 ,{0
                              1}
```

機　能　音色データを配列変数に複写します。

説　明　<音色番号>で示される音色データを、<配列名>で指定した配列変数に複写します。

複写した<配列名>の配列の一部を変更した後、VOICE命令でもとの<音色番号>に再び設定することによって音色を変更することができます。

<音色番号>……複写元音色データの音色番号を指定します。
指定範囲は、音色データがFM音源であるかPCM音源であるかによって、以下のように制限されます。

FM音源　：1～128
PCM音源：1～32

<配列名>……DIM命令で配列宣言されている文字型以外の配列を指定します。
指定は、音色がFM音源であるかPCM音源であるかによって、以下のように制限されます。

FM音源　：・48バイト以上の配列を指定します。
　　　　　・整数配列の場合、DIM配列名（23）で、大きさは48バイトです。
PCM音源：・128バイト以上の配列を指定します。
　　　　　・整数配列の場合、DIM 配列名（63）で、大きさは128バイト、要素数は64個です。

<0><1>……音源の種類を指定します。
0：FM　音色データのとき
1：PCM音色データのとき
省略すると、0を指定したとみなされます。

V

●配列には、次のようにデータが格納されます。

FM音色データの場合

	サイズ	内容	説明
+0	8	プログラムネーム	8文字までの名前
+8	4	DT1, MULT1	ディチューン，マルチプル
+12	4	TL	トータルレベル
+16	4	KS, AR	キースケール，アタックレート
+20	4	AMON, DR	ディケイレート
+24	4	SR	サスティンレート
+28	4	SL, RR	サスティンレベル，リリースレート
+32	1	FB, CNCT	フィードバック，コネクト
+33	1	LR, AMS, PMS	パン，LFO
+34	14	リザーブ	

PCM音色データの場合

	サイズ	内容	説明
+0	8	プログラムネーム	8文字までの名前
+8	8	リザーブ	
+16	2	スプリット1	サウンド1の音階の上限
+18	2	スプリット2	サウンド2の音階の上限
+30	2	スプリット8	サウンド8の音階の上限
+32	4	サウンドID1	スプリット1に使用するサウンドID
+36	4	サウンドID2	スプリット2に使用するサウンドID
+60	4	サウンドID8	スプリット8に使用するサウンドID
+64	8	エンベロープ1	スプリット1のエンベロープ
+72	8	エンベロープ2	スプリット2のエンベロープ
+120	8	エンベロープ8	スプリット8のエンベロープ

＊エンベロープデータの内容は次のとおりです.

	サイズ	内容	説明
+0	1	トータルレベル	最大音量
+1	1	アタックレート	アタックの増加レート
+2	1	ディケイレート	アタック後の減衰レート
+3	1	サスティンレベル	減衰レベル
+4	1	サスティンレート	減衰レート
+5	1	リリースレート	キーOFF時の減衰レート
+6	1	リザーブ	
+7	1	リザーブ	



PCM音色は、インスツルメントとサウンドデータで構成されていますが、VOICE COPY命令で配列に複写されるのは、このうちインスツルメントの方だけです。

関連命令 PCMREC：マイクからの音声をサンプリングして、データを配列に読み込みます。
PLAY　　：音楽の演奏を行います。

V

VOICE SET

ボイス・セット　*voice set*

音楽/音声

```
VOICE SET 配列名
```

機　能　配列に読み込まれているPCM音声データを、PLAY命令で演奏するための音色データとして設定します。

説　明　＜配列名＞……PCM音声データが読み込まれている配列を指定します。

- VOICE SET命令を実行したあと、PLAY命令でPCM音声データにメロディを付けて演奏をすることができます。
  このとき、PCM音色データの音色番号1～32は消去されるので、どの音色番号を指定しても、音声データで演奏されます。（FM音色データの音色番号は、正常に働きます。）
  PLAY命令で使用するパートは内蔵PCM音源のチャンネルに割り当てられている必要があります。
- VOICE SET命令で設定したPCM音色を、FM音源の音色と同時に演奏させるためには、PCMREC命令で録音するときに、ド（O4C）の音程で取り込む必要があります。
- VOICE SET命令実行後に、もとのPCM音色データ1～32を演奏する場合は、ファイル名を省略したLOAD@命令で、PCM音色データをCD-ROMからメモリに取り込んで復元する必要があります。

関連命令　PCMREC：マイクの音声をサンプリングして、データを配列に読み込みます。
PLAY　　：音楽の演奏を行います。

WAIT

ウェイト　*wait*　　時計

```
WAIT 時間
```

機　能　指定された時間だけ、プログラムの実行を中断します。

説　明　指定された時間だけ処理を中断して、次の命令・関数の実行を待ちます。

<時間>……時間を数値式で指定します。単位は1/100秒です。

例　題　10秒間待ってから、2つめのPRINT文を実行します。

```
1000 A$="富士通株式会社":B$="BASIC"
1100 PRINT A$
1200 WAIT 1000
1300 PRINT B$
1400 END

結果：富士通株式会社
　　　BASIC
```

W

WEND

ダブル・エンド　*wend*　　　　プログラムの分岐

```
WEND
```

機　能　WHILE～WENDループの終わりを示します。

説　明　まだWEND命令に対応付けられていない、直前のWHILE命令と対応付けられます。

例　題　キーボードから何かのキーが押されるまで、画面に時刻の表示を続けます。

```
1000 WHILE INKEY$=""
1100   LOCATE 10,10
1200   PRINT  TIME$;
1300 WEND
```

関連命令　WHILE：一連の命令を条件つきで繰り返し実行します。

W

WHILE

ホワイル *while*

プログラムの分岐

```
WHILE 式
```

機　能　一連の命令を条件付きで繰り返し実行します。

説　明　<式>が真（値が0でない）の間は、WHILE命令とそれに対応するWEND命令の間に記述された命令文を、繰り返し実行します。

<式>……繰り返しの条件となる論理式、関係式または算術式のいずれかを指定します。

- <式>の値が真でなくなると、ループ状態から抜け出し、WEND命令の次の行から実行します。
- <式>の値が最初から真でない場合は、ループは1回も行われません。
- WHILE～WENDループの内側に更にWHILE～WENDループを作成することができます。このようなループの入れ子構造は、何重にもすることができます。WHILE命令とWEND命令は、（物理的な出現順序による）対応をとって指定します。WEND命令は、まだWEND命令に対応付けられていない直前のWHILE命令と対応付けられます。ループの入れ子構造を定義する場合は、対応付けに十分注意してください。

例　題　キーボードから、何かのキーが押されるまで、画面に時刻の表示を続けます。

```
1000 WHILE INKEY$=""
1100  LOCATE 10,10
1200 PRINT  TIME$;
1300 WEND
```

関連命令　WEND　：WHILE～WENDループの終わりを示します。

FOR　：NEXT命令とループを作り、制御変数の値を変えながら、一連の命令を繰り返し実行させます。

NEXT　：FOR～NEXTループの終わりを示します。

W

WIDTH①〔省略形W.〕

ウィドス　*width*

画面表示

```
WIDTH 〔テキスト画面の一行の文字数〕〔, テキスト画面の行数〕
```

機　能　テキスト画面に表示する１行の桁数と行数を設定します。

説　明　＜テキスト画面の１行の文字数＞……BASICで扱う論理的な１行の表示文字数を、80または０の数値で指定します。

80：論理的な１行の長さをテキスト画面の１行（80桁）と同じにします。

０：論理的な１行の長さを無限大にします。すなわち、テキスト画面の各行がつながって論理的な１行の長さは無限大になります。

省略すると、現在設定されている行の文字数を指定したとみなされます。

＜テキスト画面の行数＞……１画面の行数を、20または25の数値で指定します。

- BASIC起動時には、以下のような設定がされています。
  ＜テキスト画面の１行の文字数＞：80
  ＜テキスト画面の行数＞　　　　：25
- ＜テキスト画面の１行の文字数＞に０を指定したときは、＜テキスト画面の行数＞を変更することはできません。また、０の指定は次にWIDTH 80が実行されるまで有効です。
- ＜テキスト画面の１行の文字数＞に80を指定したWIDTH命令を実行すると、以下の処理が行われます。
  - ・テキスト画面をクリアします。
  - ・CONSOLE命令で設定した項目のうち、＜PFキー表示スイッチ＞を除いた全てを解除します。
  - ・カーソルが画面の先頭行左端に移動します。

● ＜テキスト画面の１行の文字数＞に、80を指定したときと０を指定したときの違いは、次のような場合に現れます。（次ページの例題を参照してください。）
PRINT命令を実行したとき、画面に表示しようとする文字列が現在のカーソルの位置から画面の右端までの桁数より長い場合。

・WIDTH 80を指定したとき
　出力する文字列がその行に収まらないので、改行して次の行の最初の桁位置から表示を始めます。

・WIDTH 0を指定したとき
　出力する文字列を現在のカーソルの位置から表示し、文字列の途中で改行し、次の行に残りを表示します。

例　題　WIDTH 0とWIDTH 80の違いを示します。

```
1000 WIDTH 80' ★
1100 CLS
1200 COLOR 5
1300 '=== ｽｹｰﾙ表示 (10の桁) ===
1400 FOR X=0 TO 79 STEP 10
1500   LOCATE X,0
1600   PRINT RIGHT$(STR$(X ¥ 10),1);
1700 NEXT
1800 '=== ｽｹｰﾙ表示 (1の桁) ===
1900 FOR X=0 TO 79 STEP 2
2000   LOCATE X,1
2100   PRINT RIGHT$(STR$(X MOD 10),1);
2200 NEXT
2300 '=== WIDTH 80のとき ===
2400 COLOR 7
2500 LOCATE 70,2
2600 PRINT "abcde"; ' 70 桁目から5文字出す。
2700 PRINT "ABCDEF"; ' 75 桁目から6文字出す。
2800 '=== WIDTH 0 のとき ===
2900 WIDTH 0' ★
3000 LOCATE 70,4
3100 PRINT "abcde"; ' 70 桁目から5文字出す。
3200 PRINT "ABCDEF"; ' 75 桁目から6文字出す。
3300 END
```

実行結果

```
0         1         2         3         4         5         6         7
0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8 0 2 4 6 8
                                                                      abcde
ABCDEF
                                                                      abcdeABCDE
F
```

関連命令　CONSOLE　：テキスト画面のスクロールウィンドウを設定します。
SCREEN@：画面モードを変更します。

WIDTH②〔省略形W.〕

ウィドス *width*　　　　印刷

```
         ┌ デバイス名   ┐
WIDTH    ┤              ├ , サイズ
         └ #ファイル番号 ┘
```

機　能　デバイスの１行の長さを指定します。

説　明　＜デバイス名＞または＜#ファイル番号＞でデバイスを指定して、出力する１行の長さを設定します。

＜デバイス名＞……出力するデバイスとして、SCRN:、LPT0:のどちらかを、ダブルクォーテーション（"）で囲んで指定します。

＜#ファイル番号＞……出力するデバイスに割り当てられているファイル番号を指定します。指定可能なデバイスは、SCRN:、LPT0のどちらかです。
デバイスは、既にOPEN命令により出力モード（O）でオープンされていなければなりません。

＜サイズ＞　……出力する１行の長さを指定します。

●出力先のデバイスと、指定できる＜サイズ＞の関係は、以下のとおりです。

デバイス名	指定できる＜サイズ＞の範囲	BASIC起動時のサイズ
SCRN :	0 または80	80
LPT0　:	0 ～255	0

W

● "SCRN:"を指定したWIDTH命令を実行すると、以下の処理が行われます。

・テキスト画面をクリアします。

・CONSOLE命令で設定した項目のうち、<PFキー表示スイッチ>を除いた全てを解除します。

・カーソルが画面のホームポジションに移動します。

● <サイズ>に0以外の指定をすると、BASICは指定された長さの文字列の後ろに復帰改行コード（通常はCRとLF）を付加して出力を行います。

● <サイズ>に0を指定したときは、論理的な1行の長さは無限大になります。すなわち、出力結果の各行がつながって、論理的な1行を構成します。

● <サイズ>に0を指定するか、0以外を指定するかの違いは、例えば、テキスト画面では次のような場合に現れます。

PRINT命令を実行したとき、画面に表示しようとする文字列が現在のカーソルの位置から画面の右端までの桁数より長い場合。

・0を指定したとき

出力する文字列を現在のカーソルの位置から表示し、文字列の途中で改行し、次の行に残りを表示します。

・0以外（"SCRN:"では80）を指定したとき

出力する文字列がその行に収まらないので、改行して次の行の最初の桁位置から表示を始めます。

● 一度実行したWIDTH命令は、次に同じデバイスに対するWIDTH命令が実行されるまで有効です。

例　題　&が100文字からなる文字列を、1行30文字ずつプリンタに印刷します。
%が100文字からなる文字列を、1行50文字ずつプリンタに印刷します。

```
1000  OPEN "O",#1,"LPT0:"
1100  A$=STRING$(100,"&")
1200  WIDTH#1,30               '1行の長さを, 30とする.
1300  PRINT#1,A$
1400  B$=STRING$(100,"%")
1500  WIDTH#1,50               '1行の長さを, 50とする.
1600  PRINT#1,B$
1700  CLOSE#1
1800  END
```

結果：プリンタに出力される.

```
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

W

WINDOW

ウィンドウ　*window*　　　　　　　　　　　　　　　　画面表示

```
WINDOW  [ { (wx1, wy1)      }  -  { (wx2, wy2)      } ]
          { STEP (x1, y1)   }     { STEP (x2, y2)   }
```

機　能　ワールド座標内のウィンドウの範囲を指定し、ビューポートと対応付けます。

説　明　ウィンドウを、スクリーン座標上のビューポートに対応付けるため、ビューポートの左上隅と右下隅に対応するウィンドウの２つの頂点を、ワールド座標で指定します。

<wx1, wy1> ……ビューポートの左上隅の頂点に対応するウィンドウの頂点をワールド座標値で指定します。
以下の範囲で指定します。
−3.40282E+38 ～+3.40282E+38

<STEP (x1, y1) > ……最終参照座標（LP）からの距離（x1, y1）で指定します。

<wx2, wy2> ……ビューポートの右下隅の頂点に対応するウィンドウの頂点をワールド座標値で指定します。
以下の範囲で指定します。
−3.40282E+38 ～+3.40282E+38

<STEP (x2, y2) > ……最終参照座標（LP）からの距離（x2, y2）で指定します。
省略すると、現在ビューポートに設定されている座標値と同じ座標値を指定したとみなされます。

●ウィンドウの幅は、3.40282E+38以下でなければなりません。

●一度設定されたウィンドウは、次にWINDOW命令が実行されるまで有効です。

●WINDOW命令は、LPを＜wx1, wy1＞の値に設定します。

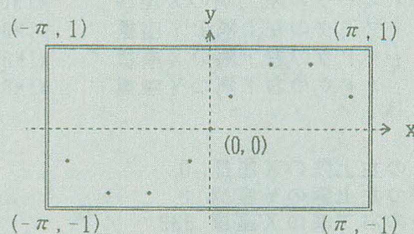
●WINDOW命令は、グラフィックを描画するための座標を設定するもので、既にVRAMに描かれているものに対しては影響を与えません。

●ウィンドウはスクリーン座標上のビューポートと対応付けられ、ウィンドウ内に描かれたものがビューポートに表示されます。
ウィンドウの範囲外に描かれたものは、表示されません。

●ウィンドウのｘ軸とｙ軸の向きを、ビューポートの正の向きに対応させる（wx1＜wx2またはwy1＜wy2の場合）ことも、負の向きに対応させる（wx1＞wx2またはwy1＞wy2の場合）こともできます。

例　題　画面の中心を原点（0, 0）とし、X軸は右が正で－π～＋π、Y軸は上が正で－1～＋1の範囲として、三角関数サイン(SIN）の値を表示します。

```
10 SCREEN@ 0 : CLS '16色モード
20 PI=3.1415926 ' π
30 WINDOW (-PI,1)-(PI,-1) ' 座標を宣言する.
40 FOR X=-PI TO PI STEP PI/5
50   PSET (X,SIN(X)),7 ' SIN(X) をプロットする.
60 NEXT
70 END
```



関連命令　WINDOW関数：ウィンドウに設定されている値を返します。
VIEW　　　　：ビューポートの範囲を指定します。

WINDOW関数

ウィンドウ関数　*window*　　画面表示

WINDOW（機能）

機　能　ウィンドウに設定されている値を返します。

説　明　WINDOW命令で設定されている、ウィンドウのワールド座標値を返します。

＜機能＞……返す座標値の種類を0～3の数値で指定します。
ウィンドウの左上隅の座標値を（wx1, wy1）、右下隅の座標値を（wx2, wy2）とすると、0～3の意味は、以下のとおりです。

0：ウィンドウの左上隅のX座標値（wx1）
1：ウィンドウの左上隅のY座標値（wy1）
2：ウィンドウの右下隅のX座標値（wx2）
3：ウィンドウの右下隅のY座標値（wy2）

例　題　ウィンドウに設定されている値を返します。

```
1000  CLS
1100  PRINT ”ウィンドウの左上隅のX座標”；WINDOW(0)
1200  PRINT ”ウィンドウの左上隅のY座標”；WINDOW(1)
1300  PRINT ”ウィンドウの右下隅のX座標”；WINDOW(2)
1400  PRINT ”ウィンドウの右下隅のY座標”；WINDOW(3)
1500  END

結果：ウィンドウの左上隅のX座標 0
      ウィンドウの左上隅のY座標 0
      ウィンドウの右下隅のX座標 639
      ウィンドウの右下隅のY座標 479
```

関連命令　WINDOW：ワールド座標内のウィンドウの範囲を指定します。
MAP関数：ワールド座標をスクリーン座標に、スクリーン座標をワールド座標に変換します。

WRITE

ライト　*write*　　　　画面表示

```
WRITE 〔式〔{ , ; }〔式〕〕…〕
```

機　能　式の結果を画面に表示します。

説　明　<式>……表示する数値または文字列を指定します。
<式>は、コンマ（,）またはセミコロン（;）で区切って複数個記述することができます。
どちらを用いても、機能上の違いはありません。
<式>を全て省略すると、改行のみが行われます。

●WRITE命令はPRINT命令と同様の処理を行います。ただし、PRINT命令とは、出力形式が異なります。WRITE命令は以下のような形式で出力を行います。
・<式>が複数個記述された場合，表示データ間にはコンマを表示します。
・<式>が文字列の場合、ダブルクォーテーション（"）で囲んで表示します。
・<式>が数値式の場合、余分な空白を表示しません。
●<式>のならびを全て出力した後は、改行を行います。

例　題　PRINT命令とWRITE命令で、同じ式を表示します。

```
1000  A$="富士通株式会社":B$="ＢＡＳＩＣ"
1100  C=12345.123#:D=.225:E=1.56235E+06
1200  PRINT A$;B$;C;D;E                   ' PRINT 命令で表示
1300  PRINT
1400  WRITE A$;B$;C;D;E                   ' WRITE 命令で表示
1500  END

結果：富士通株式会社ＢＡＳＩＣ  12345.1   .255  1.56235E+06

      "富士通株式会社","ＢＡＳＩＣ",12345.1,.255,1.56235E+06
```

関連命令　PRINT　：式の結果を画面に表示します。表示形式の点でWRITE命令とは異なります。
LPRINT：式の結果をプリンタに出力します。

WRITE#

ライト・シャープ　*write#*　　　　印刷・データファイル

```
WRITE # ファイル番号［, 式［{, | ;}〔式〕］…］
```

機　能　式の結果を、指定したファイルに出力します。

説　明　式の結果を<ファイル番号>で指定したファイルに出力します。

<ファイル番号>……OPEN時に割り当てたファイル番号を指定します。

<式>……出力する数値、または文字列を指定します。複数の<式>を指定する場合は、コンマ（,）、またはセミコロン（;）で区切ります。どちらを用いても、機能上の違いはありません。

- <ファイル番号>で指定したファイルは、出力モード（O）、または追加モード（A）で、既にオープンされている必要があります。
- WRITE#命令は、PRINT#命令と同様に式の値をファイルに出力します。出力形式の点でPRINT#命令とは異なります。
  WRITE#命令では、次のように出力します。
  - ・出力する式の結果の間にコンマ（,）を挿入します。
  - ・出力する結果が文字列の場合は、ダブルクォーテーション（"）で囲みます。
  - ・出力する結果が数値の場合は、前後の余分な余白を出力しません。
  - ・最後の<式>が出力された後に、CRとLFが出力されます。<式>を省略した場合も出力されます。

例　題　WRITE#命令を使用してデータをファイルに出力します。プリンタに文字列と数値を出力します。

```
1000  OPEN"O",#1,"LPT0:"
1100  A=123456!
1200  B$="FUJITSU"
1300  C=12.598
1400  D$="COMPUTER"
1500  WRITE #1,A,B$,C,D$
1600  CLOSE #1
1700  END

結果：プリンタに出力される.
      123456,"FUJITSU","12.598","COMPUTER"
```

関連命令　PRINT#　：　式の結果を、指定したファイルに出力します。出力形式がWRITE#命令と異なります。

W

第4章

機械語プログラム

1. 機械語プログラムの実行

BASIC用プロシジャ（機械語プログラム）はCPUの命令セット（機械語）で作成するためCPUの特性を生かした効率の良いプログラムを作成することができます。その反面、プロシジャは機械に適した言語のため、人間には理解しにくいという欠点があります。

■ プロシジャ関連命令一覧

命令	機能
CALLM	プロシジャの引数付実行。
CLEAR	スタック領域／配列領域／プロシジャ領域／DLL領域の設定。
LOADM	プロシジャをメモリへ読み込む。
PEEK	指定メモリ上のデータを読み込む。
POKE	指定メモリへデータを書き込む。
VARPTR	変数アドレスの取得。

■ プロシジャの作成方法

● 実行形式とプロシジャ

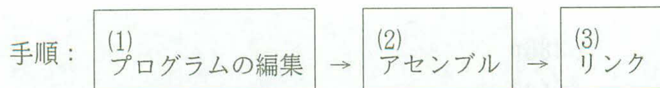
プロシジャとは、機械語で書かれたプログラムの最小単位で、機械語のret命令で終了するサブルーチンです。
実行形式とはBASICで実行可能なファイルとして、生成されたプロシジャのことをいいます。

● 開発ツール

実行形式のプロシジャを作成するには下記の開発ツールが必要です。

・アセンブリ言語　　：386ASM
・リンケージエディタ　：386LINK

● 実行形式の作成手順

手順：


開発ツール：	テキストエディタ	386ASM	386LINK
生成ファイル：	????. ASM	????. OBJ	????. REX ????. MAP

簡単なプロシジャを例に、これらの手順を説明します。

(1) プログラムの編集

テキストエディタを使用してソースプログラムを作成します。

例として、2つの数を足して結果をBASICの変数に代入するプログラムを作成します。

《プロシジャプログラム例》 (ADD.ASM)

```
1 行          .386p
2 行  param    struc
3 行           dd        ?       ;eip
4 行  result   dd        ?       ;resultを格納するoffset addr'
5 行  p1       dd        ?       ;足す数1の値
6 行  p2       dd        ?       ;足す数2の値
7 行  param    ends
8 行
9 行  CODE     segment dword public ER use32 'CODE'
10行           assume cs:CODE
11行  ADDER    proc    near
12行           mov     ebp,esp       ;edpによってparam  を参照させる
13行           mov     eax,[ebp].p1;eaxに足す数1を代入する.
14行           add     eax,[ebp].p2;eaxに足す数2を加算する.
15行           mov     ebx,[ebp].result;ebx←加算結果格納アドレス
16行           mov       [ebx],eax   ;加算結果格納アドレスに結果を格納
17行           ret                    ;プロシジャを終了し、ＢＡＳＩＣに戻る
18行  ADDER    endp
19行  CODE     ends
20行           end
```

プロシジャプログラム解説

1行　　　：386用のプログラムを記述することを宣言します。

2行～7行：足し算を行うADDERプロシジャの引数の順番を指定しています。

9行　　　：プログラム用のセグメントの指定です。必ず例と同じ指定にしてください。

10行　　　：セグメントとセグメントレジスタとの対を指定します。

11行　　　：足し算を行うADDERプロシジャの始まりを指定します。
例のように必ず、「プロシジャ名　proc near」としてください。

12行～17行：プロシジャ本体。SS,ESPレジスタ以外は破壊しても大丈夫です。
プロシジャの終了は、ret命令で終了してください。

18行　　　：プロシジャADDERの終わりを指定します。

19行　　　：プログラム用セグメントの終わりを指定します。

20行　　　：プロシジャプログラムADD.ASMの終了を指定します。

(2) アセンブル

386ASMを用いてアセンブルします。

アセンブルには、特にオプションを指定する必要はありません。

アセンブルが正しく行われると、オブジェクトファイルが作成されます。

386ASM　ADD　　← アセンブルの実行

(3) リンク

リンクには、386LINKを使用します。

386LINKは、作成されたオブジェクトファイルを実行形式に変換します。

変換元になるオブジェクトファイルは1つでも2つ以上でもかまいません。

リンクは、下記のようにリンカコマンドファイルを使用すると便利です。

????.lnk

????	←	実行形式を作成するのに必要な".OBJ"ファイル
....	←	必要に応じて複数の".OBJ"が指定出来る
−relexe	←	".REX"実行形式名の指定. 必ず指定すること

詳細は、386LINKのマニュアルを参照してください。

ADD.OBJをリンクする場合は、次のいずれかの方法によってリンクし、正常にリンクされれば、"ADD.REX"が作成されます。

1) コマンドラインから直接386LINKのパラメータを指定する場合。

```
386LINK add −relexe add
```

2）リンカコマンドファイルを使用する場合。

```
add.lnk
add
-relexe add
```

を汎用エディタで作成し、

```
386LINK @add
```

■ プロシジャの呼出方法

プロシジャを実行するには、次の３つの処理を行う必要があります。

	行わなければならない処理	該当する命令
(1)	プロシジャをメモリに読み込んで実行するための領域（プロシジャ領域）を確保する	CLEAR
(2)	(1)で確保した領域にプロシジャを読み込む	LOADM
(3)	(2)で読み込んだプロシジャを実行する	CALLM

● プロシジャ領域の確保：CLEAR

プロシジャは、BASICプログラム（プログラムテキスト）とは別の領域であるプロシジャ領域に読み込まなければなりません。

プロシジャ領域はCLEAR命令の<プロシジャ領域の大きさ>によって指定します。

BASICが起動された直後ではプロシジャ領域はありません。

概略のメモリマップは次のようになります。

BASIC起動直後（プロシジャ領域はない）	CLEARで指定した状態（プロシジャ領域を確保）
BASIC プログラム	BASIC プログラム
変数領域	変数領域
配列領域	配列領域
スタック領域	スタック領域
	プロシジャ領域

● プロシジャの読み込み：LOADM

読み込もうとするプロシジャは、LOADM文の＜ファイルディスクリプタ＞で指定します。

プロシジャを読み込む場所はLOADM文の＜オフセット＞で、プロシジャ領域の先頭からの相対位置を指定します。

● プロシジャの実行：CALLM

CALLMは、複数の引数を渡し、＜エントリオフセット＞で示されるプロシジャを実行します。

CALLMの＜引数＞は定数、変数、配列の要素などの値としてプロシジャに渡されます。

変数のアドレス、配列のアドレスを渡したい場合には、VARPTR文を使用してアドレスを取得し、それを値としてプロシジャに渡してください。

・入力I／F（CALLMからプロシジャに渡される情報）

CS	＝プロシジャ／BASICプログラム用セレクタ値
	＝Descriptor Type はEXEC READ
DS＝ES＝FS＝GS	＝プロシジャ／BASIC変数用データセレクタ値
	＝Descriptor Type は、READ WRITE
	（CS とは Aliasになっている）
SS:ESP	＝プロシジャ用スタック領域（BASICとの共有）
	プロシジャ用スタックとして256 DWORD分保証
EFLAGS	＝方向フラグ　小→大。それ以外のフラグは不定

その他のレジスタ　＝不定値

SS:ESP→ +00	return EIP	
+04	引数 1	16個まで
+08	引数 2	
	：	

・出力I／F（プロシジャからCALLMに渡される情報）

EAX　＝CALLMの戻り値

《BASICによる呼出側でのプログラム例》

```
100 CLEAR,,1024,2048,4096
110 LOADM "ADD.REX",0&
120 RESULT&=1
130 TEST&=2
140 CALLM 0&,VARPTR(RESULT&),&H12345678,TEST&
150 PRINT"RESULTは",RIGHT$("00000000"+HEX$(RESULT&), 8)
160 END
```

BASICプログラム解説

100行 ： BASICプログラムが動作するためのスタック領域、配列変数領域、プロシジャ領域の大きさを各々1024バイト、2048バイト、4096バイトに設定します。

110行 ： ADD.REXをユーザ領域のオフセット0から読み込みます。

120行 ： 正しく動作しているか確認のため、予め値を代入します。

130行 ： 足す数を指定します

140行 ： CALLM文によって110行で読み込んだプロシジャを実行します。
引数は、足し算の結果を格納する変数のアドレス、足す数1、足す数2の順で行います。

150行 : 120行で設定した値ではなく、&H1234567Aという値が表示されれば正しく動作したことになります。
160行 : 終了します。

■ プロシジャ作成の規約・注意事項

(1) プロシジャの終了は、near retによって行います。
(2) スタック領域について
・プロシジャに制御が渡った時点でのSS:ESPよりも上位のアドレスの内容は変更しないでください。
・プロシジャの使用可能なスタック領域は、256DWORDは保証するがそれ以上のスタックを必要とする場合、プロシジャは自身でスタック領域を確保する必要があまります。
・プロシジャが終了する直前(near retする)までには、SS:ESPの値をプロシジャが起動された時と同じ値にしてください。
(3) プロシジャは、CS, SS:ESP以外を破壊したままで、retしてもかまいません。
(4) 割り込みハンドラの登録はしないでください。
(5) 各種デバイス、デバイスドライバ(CON, PRN, AUXその他ファイルデバイス)やRAM, I/Oポートに対するアクセスは、基本的にF-BASIC386で処理するようにしてください。
現在のF-BASICでサポートしていないデバイスを駆動する場合は、F-BASIC386との処理のタイミングに十分注意してください。
(6) GDT, LDT, IDTの変更や80386ネイティブ/リアルモードのいずれかの割り込みテーブルの変更は行わないでください。
(7) リアルモード用プログラムは動作しません。ネイティブモードUSE16セグメントやそれで動作するプログラムも動作しません。
(8) プロシジャはCLEAR文で取得したプロシジャ領域のみを使用し、それ以外のメモリを不当にアクセスしないでください。
プロシジャ領域の他には、F-BASICからVARPTR関数によって渡された変数のアドレスをアクセスする程度にとどめてください。
また、プロシジャの存在するセグメント(セレクタ0CH, 14H)に対してセグメントリミットの変更(DOS-Extenderコール:セグメントの変更)は行わないでください。新セグメントの取得やそのセグメントに対する開放(DOS-

Extenderコールの発行を含める）も行わないでください。

(9) BASICの本体内ルーチンをコールしてもそのI／Fや動作は保証されません。

(10) プロシジャは、正常に機能するもののみ動作可能です。正常に機能しないプロシジャを動作させると、暴走してBASICに制御が戻らなくなることがあります。

(11) 数値演算プロセッサが実装されていない場合は、数値演算プロセッサ命令を発行しないでください。ハングアップします。

付　録

付録1　F－BASIC386　エラーメッセージ一覧

ERROR　1　FOR文がないのにNEXT文がありました

原因：FOR～NEXT命令が正しく使われていない。
対処：・FOR文とNEXTの数を合わせてください。
・FOR～NEXTループ内にGOTO命令やGOSUB命令がある場合は、分岐先のプログラムもチェックしてください。

ERROR　2　文法が正しくありません

原因：・プログラムの中にBASICにない命令がある。
・命令が正規の形式以外で使用されいる。
対処：・命令の綴りを修正してください。
・関数を代入文の左辺においたり、命令として使用しないでください。
・変数名は英字または漢字で始めてください。
・ERR、ERL、CSRLINの予約変数に値を代入しないでください。
・行番号は65529までの正の整数にしてください。
・関数の引数の数を合わせてください。
・DEF FNで定義した関数の使い方を確認してください。
・ラベル名の先頭は必ず＊（アスタリスク）を指定してください。

ERROR　3　GOSUB文がないのにRETURN文がありました

原因：GOSUB命令を使用しないでサブルーチンに分岐している。
対処：・GOTO命令でサブルーチンへ分岐しないようにしてください。
・メインルーチンをきちんと終わるようにしてください。
・メインルーチンが終わると自動的にサブルーチンが実行されるようになっていないかチェックしてください。

次のプログラムは誤りです。

```
        :
1000 gosub  *SUB1
        :
2000 gosub  *SUB2
        :
3000 *SUB1
        :
4000 return
```

この間にEND文がなければなりません。メインプログラムが2000行で終わっているのにEND命令がありません。そのため、2000行を実行したあと、そのまま*SUB1を実行し、4000行のRETURN命令に出会います。2000行の後にEND命令を入れると正しく終了します。

ERROR　4　DATA文のデータがないのにREAD文がありました

原因：READ命令の実行回数またはREADする変数の数よりDATAの数がなく、データが足りない。

対処：・DATA命令の形式や数を合わせてください。
・RESTORE 命令の使い方が正しいか確認してください。
・RESTORE を使用していないのに、同じDATAを読もうとしていないか確認してください。

付録

ERROR　5　関数または命令文の使い方が正しくありません

原因：パラメータの設定または値がちがう。

対処：・パラメータの範囲を合わせてください。

・パラメータの使用する数の形式が正しいか確認してください。

・宣言されていない配列や使用されていない変数を使わないようにしてください。　(GET@、PUT@ 命令等)

・指定された配列の大きさが小さくないかチェックしてください.　(GET@、PUT@ 命令等)

・配列の添字の値は正にしてください。

・PRINT USING命令の桁指定は24桁までにしてください。

ERROR　6　数値データの値が許される範囲を超えています

原因：変数型式（単精度、倍精度、整数）以上に演算結果の値が大きくなってしまった。

対処：・整数演算の結果が−32768〜+32767の範囲にあるようにしてください。

・ロング型整数演算の結果が−2147483648〜+2147483647の範囲にあるようにしてください。

・実数演算などの結果が，単精度の時は−3.40282E+38〜+3.40282E+38、倍精度の時は−1.79769313486231D+308〜+1.79769313486231D+308の範囲にあるようにしてください。

ERROR　7　メモリが足りません

原因：CLEAR命令で指定できるサイズ以上のメモリを指定した。

対処：FRE関数でとれる領域を計算して設定してください。

ERROR　8　指定された行番号が見つかりません

原因：存在しない行番号をプログラム中で指定している。

対処：GOTO、GOSUBなどの命令において、指定する行番号がプログラム中に存在するか確認してください。

ERROR　9　配列の添字の値が許される値を超えています

原因：配列の添字の値が大きすぎる。

対処：
- FOR～NEXT命令により配列の添字を変化させている場合、その値を宣言した添字以内にしてください。
- 配列の次元数を合わせてください（1次元配列に2次元指定をしないようにしてください）。
- 添字を10より大きな値に指定する場合は、DIM命令で宣言してください。

ERROR　10　重複定義を行おうとしました

原因：配列、定義関数を重複定義しようとした。

対処：
- FOR～NEXT、GOTO命令のループ内で同じ名前の配列、定義関数を定義しないでください。
- サブルーチン内で同じ名前の配列、定義関数を定義しないでください。

ERROR　11　0で割ることはできません

原因：除算の除数が0である。

対処：定義されていない変数を除数に使用しないようにしてください。

ERROR　12　直接モードでは実行できません

原因：オペレーションミス。間接モードのみ実行可能な命令を直接モードで実行しようとした。

対処：DEF FN、INPUT、LINE INPUT　などの命令は直接モードでは実行できないので、間接モードで使用してください。

ERROR　13　変数または式の型が合っていません

原因：数値を文字変数に代入しようとした。または文字を数値変数に代入しようとした。

対処：・数値を文字変数に代入しないようにしてください。
　　　・文字を数値変数に代入しないようにしてください。

次のプログラムは誤りです。

```
10 A="5"      ' ←数値変数に文字を代入している
20 print A
30 end
```

付録

ERROR 14 文字領域が足りません

対処：CLEAR命令により、配列領域を少し小さく指定してください。

ERROR 15 文字列の長さが許される範囲を超えています

原因：プログラムミスまたはオペレーションミス。文字変数に256文字以上代入しようとした。

対処：・文字変数などの演算結果が256文字以上にならないようにしてください。

・キーまたは入出力装置より1つの文字変数に256文字以上代入しないようにしてください。

ERROR 16 文字式の構造が複雑すぎます

原因：文字式の入れ子が10重を超えてしまった。

対処：文字式の入れ子の深さを10重までにしてください。

ERROR 17 続行できません

原因：オペレーションミス。BREAKキーまたはSTOP命令でプログラム停止後、プログラムの変更などを行った。

対処：停止後プログラムの変更を行うとCONT命令は実行できません。
また、停止してLIST命令やFILES命令を実行中にBREAKキーを押した後にもCONT命令は実行できません。

ERROR　18　定義されていない関数が参照されました

原因：DEF FN命令で定義されていない関数を参照しようとした。
対処：変数名の最初にFNを使用しないようにしてください。

ERROR　19　エラー処理ルーチンにRESUME文がありません

対処：エラー処理ルーチンの終りにRESUME文をいれてください。

ERROR　20　エラー処理ルーチン以外でRESUME文がありました

原因：エラーが検出されていないのにエラー処理ルーチンを実行した。
対処：・GOTOやGOSUB命令などでエラー処理ルーチンへ分岐しないようにしてください。
・メインルーチンの終了後、自動的にエラー処理ルーチンを実行していないかチェックしてください(メインルーチンとエラー処理ルーチンはEND命令によって区別するようにしてください)。

ERROR　21　定義されていないエラーコードによるエラーが発生しました

原因：ERROR命令で定義したエラー番号に誤りがあった。
対処：エラーメッセージにないエラーコード、または定義されていないエラーを発生させないでください。

ERROR　22　オペランドの記述が正しくありません

原因：オペレーションミス。

対処：・パラメータの数が正しいか確認してください。

　　　・代入文の右辺がぬけていることのないようにしてください。

ERROR　23　FOR文とNEXT文の対応が正しくありません

原因：FORの最初の判定で条件が不成立のとき対応するNEXT命令がなかった。

対処：FOR命令に対応するNEXT命令を記述してください。

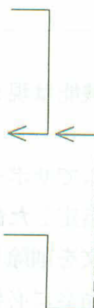
ERROR　24　WHILE文に対応するWEND文がありません

原因：WEND命令がないのにWHILE命令を実行した。

対処：２つのWHILE命令に対し１つのWEND命令で対応させたりしないようにしてください。

次のプログラムは誤りです。

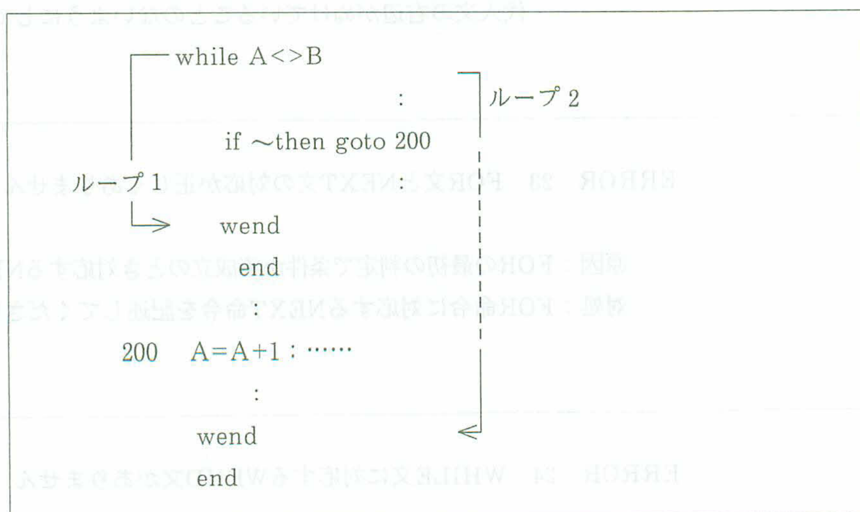
```
10 B=10:input”A”, A
20 while A<>B
30 A=A+1:print A
40 goto 70
50 wend
60 end
70 while A<>B
80 C=C+1:print C
90 goto 50
```



ERROR　25　WHILE文がないのにWEND文がありました

原因：WHILE命令がないのにWEND命令を実行した。

対処：次のようなプログラム構成になっていないかチェックしてください。



ERROR　27　システムに異常があります

原因：BASICインタプリタの内部でエラーが発生しました。

ERROR　28　この機能は現システムでは処理できません

原因：現システムでサポートしていない文を実行しようとした。
　　　または、指定した命令に必要なシステム設定がなされていない。

対処：該当する文を削除してください。
　　　または、命令に必要なシステム設定をおこなってください。

ERROR　29　定義されていないラベル名が参照されました

原因：プログラム中のステートメントで未定義のラベルを参照しようとした。
対処：ラベル名が正しいか、または参照したラベルが定義されているかどうかを確認してください。

ERROR　30　ラベル名が重複して定義されています

原因：ERROR文で指定された。

ERROR　31　IFブロックにENDIF文がありません

原因：構造化IF命令に対応するENDIF命令がない。
対処：ENDIF命令があるかどうか確認してください。

ERROR　32　IFブロックの構造が複雑すぎます

原因：IFブロックの入れ子の数が127を越えている。
対処：IFブロックの入れ子の数を確認してください。

ERROR　39　IFブロックの構成に誤りがあります

原因：IFブロックの命令の記述位置に誤りがある。
対処：構造化ブロックの構成が正しいか確認してください。

ERROR 50 ファイル番号が正しくありません

原因：指定されたファイル番号は使われていない。
対処：・ファイル番号は1から16までの範囲で指定してください。
・シーケンシャルファイルおよびランダムファイルでのファイル操作に、ファイル番号0を使わないようにしてください。

ERROR 51 ファイルモード指定が正しくありません

原因：・入力（出力）でオープンしたファイルに出力（入力）しようとした。
・MERGE命令でバイナリファイルを読もうとした。
対処：・OPEN命令の入力、出力に対して入出力命令が対応しているかチェックしてください。

例えば、 open " I", 1, "0:DATA"
print#1, "OUTPUT"

入力でオープンしたのに出力命令を実行しています。
・記述方法が正しいか確認してください。

例えば、 open " i", 1, "0:DATA"

モードは必ず大文字にしてください。

ERROR 52 指定のファイルはすでにオープンされています

原因：同じファイルを2度オープンしようとした。
対処：・すでに使用中のファイル番号を指定してファイルをオープンしないようにしてください。
・ファイルがオープンされている状態でKILL命令およびNAME命令を実行しないでください（KILLおよびNAME命令はファイルがクローズされていないと実行できません）。

ERROR　53　入出力装置に異常が発生しました

原因：・ハードエラーが発生した。
　　　・使用している媒体（フロッピィディスクなど）のキズや汚れにより使用できなくなってしまった。
　　　・ディレクトリ、FAT(file allocation table)がこわれている。
対処：・媒体を交換しても発生する場合は、別売のクリーニングディスクを使用してヘッドのクリーニングを行ってください（TownsMENUのヘッドクリーニングを使用）。
　　　・プログラムで、ディレクトリ、FATの部分には記録しないでください。
　　　・フォーマットしていないフロッピィディスクなどを使用しないでください。

ERROR　54　読み込むデータがありません。

原因：ファイルのデータをすべて読んだ後に、入力命令（INPUT#、LINE INPUT#）を実行しようとした。
対処：・シーケンシャルファイルでは、EOFを超えてINPUT#命令などを実行しないようにしてください。INPUT#などの命令の後にEOFのチェックを行ってください。
　　　・ランダムファイルではGET命令で指定したレコード番号がファイルの大きさを超えないようにしてください。LOFでレコードの最大を調べておき、それより大きな番号を指定しないようにしてください。

次のプログラムのようにしないでください。

```
10 open " I", #1, "TEST"
20 print " データファイルのリード"
30 if eof (1) then print " リードのおわり" :end
40 input#1, A$        ' ←EOFを超えて入力命令を実行しています.
50 print A$
60 goto 40
```

付録

ERROR　55　ファイルの記述に誤りがあります

原因：ファイルディスクリプタの記述に誤りがある。

対処：・ファイルディスクリプタを正しく指定してください（ドライブ名、パス、ファイル名、ファイル拡張子の記述を正しくしてください）。
・存在しないパスは指定しないでください。

ERROR　56　ファイル内に直接実行文があります

原因：オペレーションミスまたはプログラムミス。プログラムをロードするとき、行番号なしの命令がある（このエラーは、アスキー形式のプログラムをロードしているときのみ出力されます）。

対処：単なるデータファイルをまちがってロードしていないか確認してください。

ERROR　57　ファイルがオープンされていません

原因：入出力命令を実行したとき、指定したファイル番号に対するファイルがオープンされていない。

対処：・指定したファイルをオープンしてください。
・OPEN命令のファイル番号と入力（または出力）命令のファイル番号が一致しているか、確認してください。

ERROR 58 変数の型とファイル内のデータの形式が合っていません

原因：ファイル上のデータ形式と入力命令の変数の形式が異なっている。

対処：データを出力したときの変数形式と入力命令の変数形式が一致するようにしてください。

たとえば、次の例では行番号10～40の出力プログラムと行番号100～130の入力プログラムでは、変数の形式が違います（A$とA％）。

```
10 open"O", #1, "DATA"
20 A$="ABC"
30 print#1 , A$
40 close:end
```

出力プログラム

```
100 open"I", 2, "DATA"
110 if eof (2) then end
120 input#2 , A%:print A%
130 goto 110
```

入力プログラム

ERROR 59 使用中の入出力装置はオープンできません

原因：プログラムミスまたはオペレーションミス、使用中の入出力装置をオープンしようとした。

対処：使用した入出力装置はクローズしてからオープンしてください。

ERROR　60　指定の入出力装置は使用できません

原因：プログラムミスまたはオペレーションミス。入出力しようとする目的の装置が使用できる状態ではない。

対処：・指定した装置が正しく接続されているか確認してください。
・フロッピィディスクの挿入方法は正しいかチェックしてください。

ERROR　61　入出力バッファがあふれました

原因：入出力装置のバッファが処理する前にいっぱいになってしまった（このエラーはRS-232C（open"IまたはO", 1, "COM0:"）を使用時に発生する場合がある)。

対処：・データの転送速度が速すぎて、BASICのスピードではデータをバッファから取り込めないために起こるので、ボーレートを遅くしてください。
・転送速度は十分受信できる速度でも、全体のプログラムの処理速度が遅すぎると発生することがあります。

ERROR　62 プログラムが保護されています

原因：ERROR文で指定された。

ERROR　63　指定のファイルが見つかりません

原因：オペレーションミスまたはプログラムミス。存在しないファイルを指定した。

対処：・正しいファイル名を指定してください。
・ドライブ番号の指定が正しいか確認してください。

ERROR　64　指定のファイルは既に存在しています

原因：すでに存在しているファイル名で、outputモードでオープンしようとした。
対処：・古いファイルを消してもよい場合は、旧ファイルを消去してから再度登録してください。
・または、登録しようとするファイル名を新しいものに変えてください。

ERROR　65　ディスクのディレクトリ領域に空き領域がありません

原因：オペレーションミス。ディレクトリ領域がいっぱいで、新たなファイルが登録できない。
対処：不要なファイルを削除するか、または他のフロッピィディスクを使用してください。

ERROR　66　指定のファイル数を超えてファイルはオープンできません

原因：確保されているファイル数を超えてファイルをオープンしようとした。
対処：同時にオープンできるファイルの数は13までです、ファイル数を13以下にしてください。

ERROR　67　ディスクに空き領域がありません

原因：ディスクの空き領域（容量）よりおおきなデータを登録しようとした。
　　　このエラーが発生する以前にこのファイルに書かれたデータは保証されない。

対処：・不要なファイルを削除するか、または容量の残っている他のフロッピィディスクを使用してください。
　　　・空き領域は、FILES命令により知ることができます。

ERROR　68　FIELD文の変数の長さの合計が最大レコード長を超えています

原因：FIELD命令の指定がまちがっている。

対処：FIELD命令のフィールドが最大レコード長を超えないようにしてください。
　　　たとえば、最大レコード長が256バイトの場合、次の例はファイルが256バイトを越えています。

```
10 open"R", 1, "READ"
20 field#1, 20 as S$, 100 as K$, 10 as B$, 128 as E$

   20+100+10+128=258>256バイト
```

ERROR　69　FIELD文で未定義の変数にL／RSET文がありました

原因：FIELD命令で指定した文字変数、固定長文字型変数以外にLSET／RSET文で代入しようとした。

対処：LSET／RSET命令の左辺の変数を、FIELD命令で正しく定義してください。

ERROR　70　指定されたレコード番号が正しくありません

原因：存在しないレコード番号を指定した。

対処：レコード番号がまちがっていないか確認してください。

ERROR　71　ディスクのファイルの構成が正しくありません

原因：・LOAD命令でBASIC以外のプログラムをロードしようとした。
・FAT(file allocation table)がこわれている。

対処：・正しいプログラムファイルを指定してください。
・FATの部分には記録しないでください。

ERROR　72　指定されたディスク装置が使用可能な状態になっていません

原因：オペレーションミス、またはハードエラー。

対処：フロッピィディスクが正しくセットされているか確認してください。

ERROR　73　指定されたディスクは書き込みが禁止されています

原因：ディスクが書き込み禁止状態である。
対処：書き込んでもよいフロッピィディスクならば、書き込み可能状態にしてください。

ERROR　75　デバイスまたはファイルのアクセスが拒否されました

原因：オペレーションミスまたプログラムミス。次のような理由によりDOSへのファイルアクセスが拒否されました。
・許されていないデバイスやファイルのアクセスを行おうとした。
・サポートしていない媒体にアクセスしようとした。
対処：・サポートしていない媒体にアクセスしないようにしてください。
・フロッピィディスクやCD-ROMが正しくセットされているか確認してください。
・デバイス、ファイル、ディレクトリを操作する命令の形式を確認してください。

ERROR　80　テキスト領域がいっぱいでプログラムが格納できません

原因：プログラムが大きく、現在のテキストセグメント内に入らなくなってしまった。
対処：CLEAR命令により、配列領域を少し小さくしてください。または、プログラムの分割を行ってください。

付録

ERROR 81 スタック領域の大きさが足りません

原因：GOSUB、FOR～NEXT、WHILE～WEND命令が多いため、スタック領域をすべて使った。
対処：・CLEAR命令により、配列領域を少し小さくしてください。
・プログラムの分割を行ってください。
・FOR、WHILEで始まるループから、それぞれNEXT、WENDで抜けるようにしてください。
・GOSUBで呼ばれるサブルーチンが、RETURNで終わるようにしてください。

ERROR 82 単純変数領域がいっぱいになりました

原因：文字定数または単純変数が多いため、単純変数領域を使いつくしてしまった。または、RENUM命令の実行中に作業領域が不足してしまった。
対処：CLEAR命令により、配列変数領域、テキスト領域を小さく指定してください。

ERROR 83 配列変数領域がいっぱいになりました

原因：配列変数が多すぎるため、配列変数領域を使いつくしてしまった。
対処：CLEAR命令により、配列変数領域を大きく指定してください。

ERROR 84 日本語変数の定義が多すぎます

原因：日本語を含む変数名が128個を越えています。
対処：変数を整理してください。

ERROR　89　システム用作業領域がいっぱいになりました

ERROR　90　プロシジャ領域の大きさが足りません

原因：プロシジャ領域の大きさを指定しないでLOADMしようとした。
　　　または、LOADMしようとするファイルが、プロシジャ領域にはいりきらない。

対処：プロシジャ領域を大きく指定してください。

ERROR　91　DLL領域の大きさが足りません

原因：DLL領域が不足している。または、DLLのファイルが、DLL領域にはいりきらない。

対処：DLL領域を大きく指定してください。

ERROR　110　スプライトパターンが未登録です

原因：スプライトパターンを登録せずにスプライト関係の命令を実行しようとした。

対処：DEF SPRITE 0で，スプライトパターンを登録してください。

ERROR　111　スプライトキャラクタが未登録です

原因：スプライトキャラクタを登録せずにスプライト関係の命令を実行しようとした。

対処：DEF SPRITE 1で、スプライトキャラクタを登録してください。

ERROR　112　現在のモードではこの命令は使うことができません

対処：画面モードを正しく合わせてください。

付録

ERROR　115　データトラックが指定されました

原因：CD PLAYで曲でなくデータのトラックが指定された。
対処：CDをかけ替えるかまたは曲のトラックを指定してください。
Townsシステムソフトウェアや F-BASIC386のCD-ROMでは、トラック1にデータが入っています。

ERROR　120　中断しました

原因：オペレーションミス。プログラム実行中にSTOP命令が見つかったか、BREAKキーを押した。
対処：必要のないSTOP命令は取り除いてください。

ERROR　121　入力データが正しくありません

原因：数値データを入力するべきところに、文字データを入力しようとした。
対処：数値データを入力するべきところに、文字データを入力しないでください。

ERROR　124　存在しない行番号が参照されています

原因：GOTO,GOSUB文等で、存在しない行番号に分岐している。
対処：分岐先の行番号を改めてください。

ERROR　125　KILL文が指定されました　もう一度確認して下さい

原因：ファイルの「削除」を実行しようとした。

付録2　音色番号一覧表

FM音源の音色番号一覧表（1〜77）

分類	音色番号	音色	解説
金管楽器	1	トランペット TRUMPET	派手な音のトランペット．ファンファーレに向いている
	2	ホルン HORN	マイルドな音のホルン
	3	チューバ TUBA	厚みのある低音のチューバ
	4	ブラス1 BRASS 1	トランペット＋トロンボーン．ポピュラー曲のブラスセクション向き
	5	ブラス2 BRASS 2	トロンボーンの音
	6	ベル／ブラス BELL/BRASS	鐘と金管楽器が重なった音
木管楽器	7	ピッコロ PICCOLO	高音域の横笛
	8	フルート FLUTE	中音域の横笛
	9	クラリネット CLARINET	まろやかな音のクラリネット
	10	オーボエ OBOE	叙情感のあるオーボエ
	11	ファゴット FAGOTTO	オーボエに似た低音域の楽器
弦楽器	12	ストリング1 STRING 1	音域の低いストリングス．チェロ風
	13	ストリング2 STRING 2	音域の高いストリングス．バイオリン風
鍵盤楽器	14	ピアノ PIANO	一般的なアコースティックピアノの音色
	15	エレクトリックピアノ1 ELECTRIC PIANO 1	ポピュラーミュージックでよく使われるE.ピアノの音色
	16	エレクトリックピアノ2 ELECTRIC PIANO 2	E.ピアノのバリエーション
	17	エレクトリックピアノ3 ELECTRIC PIANO 3	E.ピアノのバリエーション
	18	パイプオルガン1 PIPE ORGAN 1	教会の賛美歌伴奏用パイプオルガン
	19	パイプオルガン2 PIPE ORGAN 2	大ホールの荘厳なパイプオルガン

分類	音色番号	音　　色	解　　説
鍵盤楽器	20	エレクトリックオルガン 1 ELECTRIC ORGAN1	ポピュラーミュージックでよく使われるE.オルガンの音色
	21	エレクトリックオルガン 2 ELECTRIC ORGAN 2	E.オルガンのバリエーション
	22	ハープシコード HARP-SI-CHORD	別名チェンバロ．バロック音楽によく使われる音色
	23	クラビネット CLAVINETT	ディスコ音楽によく使われる特徴のあるキーボード
ギター&ベース	24	ギター GUITAR	ジャズギター風．マイルドなタッチのギター
	25	エレクトリックベース ELECTRIC BASS 1	かためのE.ベース．ロック向きの音色
	26	エレクトリックベース 2 ELECTRIC BASS 2	深みのある音色のE.ベース．ジャズの 4 ビートに向いている
	27	シンセベース SYNTH BASS	シンセサイザ奏者が好んで使うベースの音色
シロフォン	28	シロフォン XYLOPHONE	木琴
	29	グロッケン GROCKEN	鉄琴
	30	ビブラフォン VIBRAPHONE	俗に言う「バイブ」の音．ジャズなどによく使われる
その他	31	ハープ HARP	竪琴
	32	リコーダー RECORDER	たて笛
	33	ハーモニカ HARMONICA	ハーモニカ
	34	チター ZITAR	映画「第 3 の男」のテーマで有名な弦楽器
	35	コト KOTO	代表的な和楽器の琴
ドラムス	36	スネアドラム 1 SNARE DR 1	小太鼓．ドラムセットのスネアの音
	37	スネアドラム 2 SNARE DR 2	小太鼓．マーチングドラム風の音色
	38	バスドラム BASS DR	足で踏んで鳴らす大太鼓
	39	オープンハイハット OPENED H.H	ハイハットシンバルを開いた状態で叩いた音

付録

分類	音色番号	音色	解説
ドラムス	40	クローズハイハット CLOSED H.H	ハイハットシンバルを閉じた状態で叩いた音
	41	シンバル CYMBAL	トップシンバルの音. 4ビートジャズに向いている
パーカッション	42	ティンパニ TIMPANI	低音域の打楽器. O2の音域で使用
	43	カウベル COW BELL	牛の首に付けるベルの形をしたパーカッション
	44	ベル1 BELL 1	ベル
	45	ベル2 BELL 2	チューブラベル
	46	スティールドラム1 STEEL DR 1	ドラム罐を叩いて弾く打楽器
	47	スティールドラム2 STEEL DR 2	スティールドラムのバリエーション
その他	48	シンセブラス SYNBRASS	派手な感じのシンセブラス
	49	シンセリード1 SYNLEAD 1	うねりをもったムーグ風の音
	50	シンセリード2 SYNLEAD 2	ファニーな感じの音
	51	シンセリード3 SYNLEAD 3	5度音程を同時に出す.
	52	シンセリード4 SYNLEAD 4	高域のきれいなリード音
	53	シンセリード5 SYNLEAD 5	強いクセのある音.
	54	シンセサイザークラビネット SYN CLAVI	ファンク系にあう音.
	55	シンセサイザーストリング SYNSTRG	ソリーナ風の音色
	56	シンセサイザーマリンバ SYNMARI	シンセサイザーマリンバ
	57	シンセサイザーベル SYNBELL	エフェクト的に使用
	58	エクスパンドベル EXP. BELL	ベルと伸びる音の合成音
	59	ウッドブロック WOOD BLK	パーカッションの一種. アクセントを加えるのに使用

分類	音色番号	音　色	解　説
その他	60	カスタネット CASTANET	フラメンコを踊りながら叩く楽器
	61	PWM シンセ PWM SYN	PWM(パルスワイズモジュレーション)を使ったアナログシンセ
	62	スクールチャイム SC. CHIME	学校で使うチャイムの音
	63	ウォーターグラス WAT GLASS	水の入ったコップを叩いた音
	64	チューブラベル TUBEBELL	チューブラベル 2
	65	ベルオルガン BELL ORG	ベルとオルガンの合成音
	66	リバースエコーベル REVEBELL	リバースエコー効果がかかったベル
	67	エクスプロージョン EXPLOSIO	爆発音
	68	MUOWEEEN	擬声音 1
	69	BONWAHHH	擬声音 2
	70	トイピアノ TOYPIANO	おもちゃのピアノ
	71	ラバーバンド RUB BAND	輪ゴムをはじいた音
	72	バッドチューニングラジオ BT. RADIO	壊れたラジオ
	73	スター STAR	流れ星の流れる音
	74	カンパイ KANPAI	ワイングラスの乾杯音
	75	JIS キーボード JIS KEYB	パソコンのキーボード音
	76	ハンドクラップ HANDCLAP	手拍子の音
	77	サインウェイブ SINE WAVE	正弦波の音．チューニング用に使用

■FM音源の音色番号一覧表(78〜128)

音色番号	音　色	音色番号	音　色
78	TRPT2	104	CELESTA
79	TRPT3	105	ORGAN
80	FLUGHORN	106	E.ORG3
81	HORN2	107	P.ORG3
82	HORN3	108	E.GUITAR
83	HORN4	109	BANJO
84	BRASS3	110	GUITAR2
85	BRASS4	111	GUITAR3
86	BRASS5	112	SITAR
87	FLUTE2	113	E.BASS3
88	PANFLUTE	114	A.BASS1
89	SAX1	115	A.BASS2
90	SAX2	116	HARMON12
91	STRG3	117	MARIMBA1
92	STRG4	118	MARIMBA2
93	STRG5	119	VIBE2
94	STRG6	120	KOTO2
95	STRG7	121	SD&RIM
96	PIANO2	122	SD3
97	PIANO3	123	BD2
98	PIANO4	124	BD3
99	PIANO5	125	CYMBAL2
100	HONKY	126	TOMTOM1
101	HARPSIC2	127	TOMTOM2
102	HARPSIC3	128	TOMTOM3
103	HARPSIC4		

付録

付録3　キャラクタコード表

下位＼上位	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		DE	(space)	0	@	P	'	p				ー	タ	ミ		
1	SH	D1	!	1	A	Q	a	q			。	ア	チ	ム		
2	SX	D2	"	2	B	R	b	r			「	イ	ツ	メ		
3	EX	D3	#	3	C	S	c	s			」	ウ	テ	モ		
4	ET	D4	$	4	D	T	d	t			、	エ	ト	ヤ		
5	EQ	NK	%	5	E	U	e	u			・	オ	ナ	ユ		
6	AK	SN	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
7	BL	EB	'	7	G	W	g	w			ァ	キ	ヌ	ラ		
8	BS	CN	(	8	H	X	h	x			ィ	ク	ネ	リ		
9	HT	EM	)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
A	LF	SB	*	:	J	Z	j	z			ェ	コ	ハ	レ		
B	HM	EC	+	;	K	〔	k	{			ォ	サ	ヒ	ロ		
C	CL	→	,	<	L	¥	l	|			ャ	シ	フ	ワ		
D	CR	←	−	=	M	〕	m	}			ュ	ス	ヘ	ン		
E	SO	↑	.	>	N	∧	n	—			ョ	セ	ホ	゛		
F	SI	↓	/	?	O	−	o	DL			ッ	ソ	マ	゜		

付録 4　数学関数

ここに示す数学関数は、BASICでは組み込み関数として用意されていませんが、用意されているいくつかの関数を用いて求めることができます。

以下にその数式を示しますので参考にしてください。

数　学　関　数	等　価　な　式
SEC(X):セカント	1/COS(X)
CSC(X):コセカント	1/SIN(X)
COT(X):コタンジェント	1/TAN(X)
ARCSIN(X):アークサイン	ATN(X/SQR(1−X*X))
ARCCOS(X):アークコサイン	1.5707963267948966−ATN(X/SQR(1−X*X))
ARCSEC(X):アークセカント	ATN(SQR(X*X−1))+(X<0)*3.1415926535897932
ARCCSC(X):アークコセカント	ATN(1/SQR(X*X−1))+(X<0)*3.1415926535897932
ARCCOT(X):アークコタンジェント	1.570796−ATN(X)
SINH(X):ハイパーボリックサイン	(EXP(X)−EXP(−X))/2
COSH(X):ハイパーボリックコサイン	(EXP(X)+EXP(−X))/2
TANH(X):ハイパーボリックタンジェント	(EXP(X)−EXP(−X))/(EXP(X)+EXP(−X))
SECH(X):ハイパーボリックセカント	2/(EXP(X)−EXP(−X))
CSCH(X):ハイパーボリックコセカント	2/(EXP(X)−EXP(−X))
COTH(X):ハイパーボリックコタンジェント	(EXP(X)+EXP(−X))/(EXP(X)−EXP(−X))
ARCSINH(X):ハイパーボリックアークサイン	LOG(X+SQR(X*X+1))
ARCCOSH(X):ハイパーボリックアークコサイン	LOG(X+SQR(X*X−1))
ARCTANH(X):ハイパーボリックアークタンジェント	LOG((1−X)/(1+X))/2
ARCSECH(X):ハイパーボリックアークセカント	LOG((1+SQR(1−X*X))/X)
ARCCSCH(X):ハイパーボリックアークコセカント	LOG((1+SNG(X)*SQR(1+X*X))/X)
ARCCOTH(X):ハイパーボリックアークコタンジェント	LOG((X+1)/(X−1))/2

付録5　ESCシーケンス

網がけは無視されることを示します.

シーケンス	略　称	機能・意味
ESC［ P1；Pc H もしくは ESC［ P1；Pc f	CUP HVP	カーソルをP1行Pc桁へ移動します. 各数値パラメータのデフォルト値は1です. 行もしくは桁の指定が画面サイズを越えた場合，画面の下端（システム行を除く）もしくは右端への移動とみなします.
ESC［ Pn A	CUU	カーソルを同一桁のまま上へPn行移動します．デフォルト値は1です. ただし，画面の上端より上へのカーソル移動は無視され，カーソルは上端に移動します.
ESC［ Pn B	CUD	カーソルを同一桁のまま下へPn行移動します．デフォルト値は1です. ただし，画面の下端（システム行は除く）より下へのカーソル移動は無視され，カーソルは下端に移動します.
ESC［ Pn C	CUF	カーソルを同一行のまま右へPn桁移動します．デフォルト値は1です. ただし，画面の右端より右へのカーソル移動は無視され，カーソルは右端に移動します.
ESC［ Pn D	CUF	カーソルを同一行のまま左へPn桁移動します．デフォルト値は1です. ただし，画面の左端より左へのカーソル移動は無視され，カーソルは左端に移動します.
ESC［ 6n	DSR	キーボードバッファの先頭に下記のCPRシーケンスが挿入されます.
ESC［ P1；Pc R	CPR	このシーケンスは，上記のDSRシーケンスによりキーボードバッファの先頭（読み込みポインタの前）に挿入されるもので，カーソル位置（P1行，Pc桁）を通知します.
ESC［ s	SCP	現在のカーソル位置が保存されます. このカーソル位置は下記のRCPシーケンスにて回復する際の値です.
ESC［ u	RCP	上記のSCPシーケンスにて保存されたカーソル位置にカーソルを移動します.
ESC［　Ps J	ED	以下に示すようにPsの値に従ってディスプレイ画面を消去します. （Ps=0～2：デフォルト値は0です） 0：カーソル位置から画面の最後まで（システム行を除く）を消去します．カーソル位置は変わりません. 1：先頭行の左端からカーソルの位置までを消去します．カーソル位置は変わりません. 2：文字ディスプレイ画面全体（システム行を除く）を消去し，カーソルを画面の先頭行先頭桁位置（HOME位置）に移動します.

網がけは無視されることを示します.

シーケンス	略　称	機能・意味
ESC [Ps K	EL	以下に示すようにPsの値に従ってディスプレイ画面を消去します．この際，カーソル位置は変わりません．(Ps=0～2：デフォルト値は0です). 0：カーソル位置からその行の終わりまで消去します. 1：行の左端からカーソルの位置までを消去します. 2：カーソルが位置している行を消去します.
ESC [= Ps h	SM	以下に示すようにPsの値に従って画面の表示モードを設定します. (Ps=0～3：デフォルト値は0です) 0：40桁 × 25行 1：40桁 × 20行 2：80桁 × 25行 3：80桁 × 20行 画面全体（システム行を含む，ただしモード表示領域を除く）を消去しカーソル位置を画面の先頭行先頭桁位置（HOME位置）に移動します.
ESC [= Ps h もしくは ESC [? 7 h		(Ps=7) 行の右端に文字を出力した際，次の行の先頭にカーソル移動させるモードとなります. 文字ディスプレイの初期化時にはこのモードが設定されています.
ESC [= Ps l	RM	以下に示すようにPsの値に従って画面の表示モードを設定します. (Ps=0～3：デフォルト値は0です) 0：40桁 × 25行 1：40桁 × 20行 2：80桁 × 25行 3：80桁 × 20行 画面全体（システム行を含む，ただしモード表示領域を除く）を消去しカーソル位置を画面の先頭行先頭桁位置（HOME位置）に移動します.
ESC [= Ps l もしくは ESC [? 7 l		(PS=7) 行の右端に文字を出力した際，カーソル位置が変更しないモードにします.

付録

網がけは無視されることを示します.

シーケンス	略　称	機能・意味
ESC［ Ps；…… ；Ps m	SGR	以降に続く文字のアトリビュートを複数のパラメータにて指定します. 各数値パラメータのデフォルト値は0です. 0：すべてのアトリビュートを解除 1：高輝度 2：バーチカルライン 3：オーバーライン 4：アンダスコア 5：ブリンク 6：アンダライン 7：リバース 8：シークレット 30：表示色　黒色 31：　赤色 32：　緑色 33：　黄色 34：　青色 35：　紫色 36：　水色 37：　白色 40：背景色　黒色 41：　赤色 42：　緑色 43：　黄色 44：　青色 45：　紫色 46：　水色 47：　白色 ただし，システムでサポートされていないものは無動作となります.（網がけの部分） 背景色を指定すると画面全体の背景色が変わります.
ESC［Ps v		カーソルの表示をPsの値に従い制御します. （Ps=0〜1：デフォルト値は0です） 0：表示，1：非表示

網がけは無視されることを示します.

シーケンス	機能・意味
ESC［ Pn ; …… 　　　　; Pn p もしくは ESC［ ”string” ;p もしくは ESC［ Pn ; ”string” ; 　　………　　; Pn p もしくは その他のストリングと10進数の組み合わせ	第1番目の文字コードに第2番目の文字コード列を割り当てます. ただし，第1番目の文字が0の場合，次の文字コードと合わせてPFキーの文字コードである80xxHの下位バイト:xxHを表します. すなわち，PF 1キーに文字’A’を割り当てる場合以下のシーケンスとなります. 　ESC［0 ; 1 ; ”A” p 最大割り当て文字数は以下のとおりです. ・PFキー　: 15文字 ・編集キー　: 7文字 ・コントロールキー以外の文字キー　: 7文字 ・コントロールキー（CTRL + @～_）: 割り当て不可
ESC = row clmn	カーソルをrow行clmn行に移動させます. ただしrow，clmnには，行あるいは桁の値に1fHを加えた文字コード(カーソル座標コード)を使用します．すなわち1行目ならばスペース(20H)です. なお，行もしくは桁の指定が画面サイズを越えた場合下端(システム行を除く)もしくは右端への移動とみなします.
ESC 1	現在のカーソル位置にタブを設定します.
ESC 2	現在のカーソル位置に設定されているタブを解除します.
ESC 3	設定されているすべてのタブを解除します.
ESC #	キーボードからの入力を禁止します.
ESC ”	キーボードからの入力禁止を解除します.
ESC *	画面全体(システム行を除く)を消去し，カーソルを画面の先頭行先頭桁位置(HOME位置)に移動します.
ESC Y	カーソル位置から画面の最後まで(システム行を除く)を消去します．カーソル位置は変わりません.
ESC T	カーソル位置からその行の終わりまでを消去します．カーソル位置は変わりません.
ESC E	カーソル位置の行に空白(20H)行(アトリビュートはデフォルトを設定)を挿入し，カーソルをその行の先頭に移動します. 行挿入前のカーソル位置の行以降は1行下へ移動(スクロールダウン)し，最終行は失われます.
ESC R	カーソル位置の行を削除し，以降の行を上へ1行移動(スクロールアップ)します. 最終行は空白(20H)とデフォルトアトリビュートが書き込まれます. カーソル位置は変わりません.

網がけは無視されることを示します．

シーケンス	機能・意味
ESC ?	キーボードインタフェースを通してカーソル位置を行桁およびCRコード(0DH)の順に通知します．行桁の値には1FHを加えた文字コード(カーソル座標コード)を使用します．
ESC X speed	スクロールスピードを指定します．
ESC . ca1 ca2	以下のようなca1，ca2のパラメータに従ったカーソルの属性を設定します． ca1　7 6 5 4 3 2 1 0 カーソルスタートラスタ カーソルブリンク 00：ブリンクしない 01：カーソル非表示 10：高速ブリンク 11：低速ブリンク カーソルサイズ 0：半角 1：全角 ca2　7 6 5 4 3 2 1 0 カーソルエンドラスタ
カーソルスタートとカーソルエンドラスタは，その範囲に囲まれたカーソルを指定します．	
ESC G at	以降に続く文字のアトリビュート(at)すなわちデフォルトアトリビュートを指定します． at　7 6 5 4 3 2 1 0 色識別番号(0〜7) リバース ブリンク 高輝度

付録６　インタプリタとコンパイラの機能の違い

命令	インタプリタ	コンパイラ
CALLM	引数を16個まで指定できます。	指定できる引数の数に制限はありません。
CHAIN	①項目として、＜MERGE＞、＜行番号式＞、＜ALL＞、＜DELETE範囲＞を指定できます。 ②＜MERGE＞指定がないと、それまでのプログラムで使用していたファイルはクローズされます。	①項目として指定できるのは、＜ALL＞だけです。 ②＜MERGE＞指定はありませんが、それまでのプログラムで使用していたファイルはクローズされません。 ③プログラム中に複数のCHAIN命令があるとき、１つのCHAINで＜ALL＞を指定したら、そのプログラム中の他のCHAIN命令でも＜ALL＞を指定する必要があります。
CLEAR	全ての変数を初期化し、スタック領域、配列変数領域、およびプロシジャ領域の大きさを指定します。DEF命令により定義された全ての情報が無効になります。	プロシジャ領域の大きさを指定します。それ以外の指定は無視され、他の領域および変数には影響しません。
COMMON	CHAIN命令で連結されるプログラムの、呼び出し元と呼び出し先で、異なる内容のCOMMON命令を記述することができます。	CHAIN命令で連結されるプログラムの、呼び出し元と呼び出し先の両方のCOMMON命令の内容が、同じでなければなりません。
CONT	使用できます。	使用できません。CONT命令を含むプログラムをコンパイルすると、エラーになります。
DEF FN	①実行順序にしたがって、宣言が有効になります。（注） ②関数の＜名前＞は、半角文字なら40文字以内、全角文字なら20文字以内で指定します。	①ソースプログラム上の出現順序で宣言が有効になります。ソースプログラム上の出現順序と、プログラムの実行順序が異なる場合には、注意が必要です。（注） ②関数の＜名前＞の長さに制限はありません。
DEFINT DEFLNG DEFSNG DEFDBL DEFSTR	実行順序にしたがって、型宣言が有効になります。（注）	ソースプログラム上の出現順序で型宣言が有効になります。ソースプログラム上の出現順序と、プログラムの実行順序が異なる場合には、注意が必要です。（注）
DELETE	使用できます。	使用できません。DELETE命令を含むプログラムをコンパイルすると、エラーになります。
ERL関数	READ命令の実行でエラーが発生したときに、DATA命令の行番号を返します。	READ命令の実行でエラーが発生したときに、READ命令の行番号を返します。
FRE関数	機能の指定により、３種類の未使用領域の大きさを返します。	機能の指定に関わりなく、BASICで使用可能な未使用領域領域の大きさを返します。

（注）● DEF命令の機能の違い（p.517）参照。

命令	インタプリタ	コンパイラ
LIST LLIST LOAD MERGE	使用できます。	使用できません。これらの命令を含むプログラムをコンパイルすると、エラーになります。
MOUSE関数	式を記述することができます。	式を記述することはできません。定数のみ記述できます。
NEW RENUM	使用できます。	使用できません。 NEW命令、RENUM命令を含むプログラムをコンパイルすると、エラーになります。
RUN	①形式1（メモリ内のプログラムの実行）が使用できます。 ②形式2（ディスク上のプログラムを読み込んで実行）が使用できます。 ・＜R＞指定ができます。 ・指定するプログラムは、アスキー形式またはバイナリ形式で保存されたBASICのソースプログラムでなければなりません。	①形式1（メモリ内のプログラムの実行）は使用できません。 ②形式2（ディスク上のプログラムを読み込んで実行）が使用できます。 ・＜R＞指定はできません。 ・指定するプログラムは、コンパイルずみの実行形式プログラムでなければなりません。
SAVE	使用できます。	使用できません。SAVE命令を含むプログラムをコンパイルすると、エラーになります。
STOP	①プログラムの処理を中断してBASICエディタに戻ります。 ②ファイルのクローズは行われません。 ③CONT命令で中断された次の命令から実行を再開することができます。	①プログラムの処理を中断してメッセージを表示し、キー入力により終了します。 ②ファイルのクローズが行われます。 ③CONT命令で実行を再開することはできません。
TRON	プログラム中に記述されている全ての行について、実行した行の行番号を表示していきます。	ソースプログラム中の非実行文については、行番号は表示されません。ただし、非実行文と実行文が同一行に存在するときは、行番号が表示されます。

上記の命令の他に、インタプリタとコンパイラの機能は以下の点が異なります。

● 変数名

インタプリタでは、日本語を含む変数名を128個まで使用できます。また、変数名は最初の40文字で区別します。

コンパイラでは、日本語を含む変数名の個数に制限はありません。また、変数名はすべての文字で区別します。

● 処理速度

インタプリタと比べて、コンパイラは非常に速くプログラムを実行します。このため、FOR～NEXTループなどをタイミングをとるために使用している場合には、値の変更が必要になります。

● DEF命令の機能の違い

次のDEF命令は、インタプリタとコンパイラで機能が異なります。

DEF FN、DEFINT、DEFLNG、DEFSNG、DEFDBL、DEFSTR

インタプリタは、実行順序にしたがって、DEF命令による宣言が有効になります。これに対し、コンパイラでは、実行順序に関係なく、ソースプログラムをコンパイルしたときの出現順序にしたがって、DEF命令による宣言が有効になります。

〔プログラム例〕

```
10  DEFINT X
20  GOTO 80
30  X=32767+1
40  PRINT X
50  STOP
60  END
70  '
80  ' INITIALIZE
90  '
100 DEFSNG X
110 GOTO 30
```

〔説明〕

インタプリタでは、10行で整数型の変数として宣言されたXを、100行で単精度の変数に変更した後、30行の代入文を実行します。したがって、30行を実行するときのXは単精度の変数なので、32768の値を代入することができます。

コンパイラでは、実行順序に関係なく、10行～90行のXは整数型とみなし、100行～110行のXは単精度とみなします。したがって、30行の代入文を実行するときのXは整数型となり、32767を超える値は代入できないため、エラーとなります。

付録7　F−BASIC 386 互換表

注）○：命令有り
△：他の機能で代替
×：〃 なし
−：他の命令に包含

命　令	形　式
ABS関数	ABS（式）
AKCNV$関数	AKCNV$（文字式）
ASC関数	ASC（文字式）
ATN関数	ATN（式）
AUTO	AUTO [行番号] [,[増分値] [, REM]]
BEEP	BEEP [{0|1}]
BGM	BGM {0|1}
CALL	CALL　変数名 [(引数 [, 引数] …)]
CALLM　形式1 形式2	CALLM　オフセット [, 引数] … 戻り値＝CALLM（エントリオフセット [, 引数] …)
CALLS　形式1 形式2	CALLS ["ファイルディスクリプタ"] [, 実パラメータ] CALLS　"フィイルディスクリプタ", 実パラメータブロック名
CDBL関数	CDBL（式）
CD CONT	CD CONT
CDINF	CDINF　配列名
CD PAUSE	CD PAUSE
CD PLAY	CD PLAY [{開始トラック番号 [, 終了トラック番号] | (開始分, 開始秒, フレーム) [−(終了分, 終了秒, フレーム)}]
CDSTAT関数	CDSTAT　配列名
CD STOP	CD STOP
CDSTIME$	CDSTIME$（トラック番号）
CHAIN	CHAIN [MERGE] "ファイルディスクリプタ" [,[行番号式] [, ALL] [, DELETE　範囲]
CHR$	CHR$（式 [, 式] …）
CINT関数	CINT（式）
CIRCLE　形式1	CIRCLE [@] (x, y), 半径 [,[色] [,[比率]] [,[開始位置] [, 終了位置] [,[{F|N}] [, 論理操作]]]]]

386　→F-BASIC386 V2.1L10
GEAR→GEAR BASIC
V3.4　→F-BASIC V3.4
86HG　→F-BASIC86HG V1.2
86　→F-BASIC86 V3.1

386	GEAR	V3.4	86HG	86	機　能
○	○	○	○	○	式の絶対値を返す.
○	○	○	○	○	文字列の中のANK文字を日本語文字に変換する.
○	○	○	○	○	最初の文字のキャラクタコードを返す.
○	○	○	○	○	アークタンジェントの値を返す.
○	×	○	○	○	行番号を自動的に発生する.
○	×	○	○	○	内蔵スピーカによりブザーを鳴らす.
○	○	○	×	×	並列動作（音楽演奏をしながら，別の命令を行う）のコントロールをする.
×	×	○	○	○	機械語サブルーチンを呼び出す.
○	○	×	×	×	機械語プログラムを呼び出して実行する. 機械語プログラムを実行し，戻り値を返す.
×	×	×	○	×	サブプログラムの呼び出しを行う.
○	○	○	○	○	倍精度実数値に型変換する.
○	○	×	×	×	CD演奏を再開する.
○	○	×	×	×	CDの内容を返す.
○	○	×	×	×	CD演奏を一時停止する.
○	○	×	×	×	CD演奏を開始する.
○	○	×	×	×	CDドライブの状態を返す.
○	○	×	×	×	CD演奏を停止する.
○	○	×	×	×	CDの各曲の開始時間を返す.
○	×	○	○	○	指定したファイルのプログラムへ変数を引き渡し，実行する.
○	○	○＊	○	○	式の値をキャラクタコードとするANK文字を返す． ＊ただし，F-BASEC V3.4では式は1個しか書けない.
○	○	○	○	○	整数値に変換する.
−	−	○	−	−	画面上の任意の座標を中心点として，円または円弧を描く.

命　　令	形　　式
形式 2	CIRCLE [@] {(wx, wy) | STEP (x, y)}, 半径 [, [色 1] [, [比率] [, [開始位置] [, [終了位置] [, [{F|N}] [, 論理操作] [, {色 2 | タイルストリング}]]]]]]]
形式 3	CIRCLE {(wx, wy) | STEP (x, y)}, 半径 [, [色 1] [, [比率] [, [開始位置] [, [終了位置] [, [{F | N}] [, [論理操作] [, {色 2 | タイルストリング | ラインスタイル}]]]]]]]
形式 4	CIRCLE {(wx1, wy1) | STEP (x1, y1)} − {(wx2, wy2) | STEP (x2, y2)} − {(wx3, wy3) | STEP (x3, y3)} [, [色] [, 論理操作] [, ラインスタイル]]]
CLEAR 形式 1	CLEAR[, , [スタック領域の大きさ] [, [配列変数領域の大きさ] [, [プロシジャ領域の大きさ] [, DLL領域の大きさ]]]]
形式 2	CLEAR [, [BASICの使用する上限のメモリアドレス] [, スタック領域の大きさ]]
形式 3	CLEAR [, [メモリの上限] [, スタック領域の大きさ] [, [配列変数領域の大きさ] [, テキスト領域の大きさ]]]]
CLNG	CLNG (式)
CLOSE	CLOSE [[#] ファイル番号 [, [#] ファイル番号] …]
CLRF	CLRF 部品名
CLS	CLS [消去範囲コード]
COLOR 形式 1 (COL.) 形式 2	COLOR [文字色] [, [背景色] [, [前景色]]]] COLOR [文字色] [, [背景色] [, [前景色] [, アトリビュート]]]]
COLOR=	COLOR= (指定色, 標示色)
COLOR@	COLOR@ (x1, y1) − (x2, y2) [, [テキストカラー] [, アトリビュート]]
COMMON	COMMON 変数名 [, 変数名] …
COM (n) ON／OFF／STOP	COM (ポート番号) {ON|OFF|STOP}
CONNECT 形式 1	CONNECT (x1, y1) − (x2, y2) [− (x3, y3)] … [, [色] [, 論理操作]]
形式 2	CONNECT [{(wx1, wy1) | STEP (x1, y1)}] − {(wx2, wy2) | STEP (x2, y2)} [− {(wx3, wy3) | STEP (x3, y3)}] … [, [色] [, 論理操作]]

386	GEAR	V3.4	86HG	86	機　　　能
–	–	×	○	○	画面上の任意の座標を中心点として，円または円弧を描く．
○	○	×	×	×	
○	○	×	×	×	指定した3点を通る円または円弧を描く．
○	○	×	×	×	すべての数値変数を0に，すべての文字変数を空文字列に初期設定し，オペランドの指定によりBASICの使用する領域を設定する．
×	×	○	×	×	
×	×	×	○	○	
○	○	×	×	×	ロング型整数値に変換する．
○	○	○	○	○	ファイルのクローズ処理を行う．
×	○	×	×	×	部品内のテキストデータを消す．
○	×	○	○	○	指定画面をクリアする．
–	×	○	–	–	テキスト画面，グラフィック画面の表示色，背景色を指定する．
○	×	×	○	○	
×	×	○	○	○	パレットコードを指定する． 注）V3.4　⇨8色中8色（4096色，26万色モードでは使用不可） 86HG／86⇨16色中8色
×	×	×	○	○	テキスト画面に書かれた文字に色や機能を設定する．
○	×	○	○	○	CHAIN命令により連結されるプログラムへ引き渡す変数を指定する．
○	×	○	○	○	通信回線からの割り込みを許可，禁止，停止します．
–	×	○	–	–	指定座標間を直線で結ぶ．
–	×	×	○	○	

付録

命　　令	形　　式
形式3	CONNECT [{(wx1, wy1) | STEP (x1, y1)}] − {(wx2, wy2) | STEP (x2, y2)} [− {(wx3, wy3) | STEP ((x3, y3)}] … [, [色] [, [論理操作] [, {N [, ラインスタイル] | F [, {中塗り色 | タイルストリング}]}]]]]
CONSOLE　形式1 形式2	CONSOLE [スクロール開始行] [, [スクロール行数] [, [PFキー表示スイッチ]]] CONSOLE [スクロール開始行] [, スクロール行数] [, [PFキー表示スイッチ] [, [コンソールカラースイッチ] [, [ひらがなスイッチ]]]]]
CONT (C.)	CONT
COS関数	COS (式)
CSNG関数	CSNG (式)
CSRLIN	CSRLIN
CVD関数	CVD (8バイトの文字列)
CVDMBF関数	CVDMBF (8バイトの文字列)
CVI関数	CVI (2バイトの文字列)
CVL関数	CVL (4バイトの文字列)
CVS関数	CVS (4バイトの文字列)
CVSMBF関数	CVSMBF (4バイトの文字列)
DATA	DATA　定数 [, 定数] …
DATE	DATE
DATE$	DATE $ [=" yy／mm／dd"]
DEBUG ON/OFF	DEBUG {ON|OFF} [, N]
DEFDBL	DEFDBL　文字の範囲 [, 文字の範囲] …
DEF FN	DEF FN名前 [(パラメータリスト)] =式
DEF FONT	DEF FONT　"フォント名"
DEFINT	DEFINT　文字の範囲 [, 文字の範囲] …
DEF KANJI	DEF KANJI　外字コード, ドットパターン

386	GEAR	V3.4	86HG	86	機能
○	×	×	×	×	指定座標間を直線で結ぶ.
○ ×	× ×	− ○	○ ×	○ ×	スクロールウィンドウの大きさを設定する.
○	×	○	○	○	BREAKキー入力／STOP命令実行によって停止しているプログラムの実行を再開する.
○	○	○	○	○	コサインの値を返す.
○	○	○	○	○	単精度実数値に変換する.
○	×	○	○	○	画面上のカーソルの垂直位置を返す.
○	○	○	○	○	8バイトの文字列を倍精度の実数に変換する.
○	×	×	×	×	マイクロソフト内部形式の文字で表現された値を，倍精度の実数に変換する.
○	○	○	○	○	2バイトの文字列を整数に変換する.
○	○	×	×	×	4バイトの文字列をロング型整数に変換する.
○	○	○	○	○	4バイトの文字列を単精度の数値に変換する.
○	×	×	×	×	マイクロソフト内部形式の文字で表現された値を，単精度の実数に変換する.
○	○	○	○	○	READ命令によって読み込まれる数値，文字定数を指定する.
○	○	○	○	○	1月1日を基準とする実行当日までのトータル日数を返す.
○	○	○	○	○	内蔵タイマーの示す日付を返すかまたは変更する.
×	×	×	○	×	会話型のデバッグ機能使用可能状態の設定と解除を行う.
○	○	○	○	○	変数の型を倍精度実数に宣言する.
○	×	○	○	○	ユーザの関数を定義し，それに名前をつける.
○	×	×	×	×	SYMBOL命令の文字フォントを指定する.
○	○	○	○	○	変数の型を整数に宣言する.
○	○	○	○	○	利用者定義文字の定義を行う. 注）外字コードはV3.4では　&H7521～&H757E, 86HG／86では&H7521～&H757E, &H7621～ &H767Eが使用可.

命　　令	形　　式
DEF LEN	DEF LEN 文字列の長さ　AS　文字変数名［文字列の長さ　AS　文字変数名］
DEFLNG	DEFLNG　文字の範囲［，文字の範囲］
DEF MOVIE　形式1 形式2 形式3	DEF MOVIE　1，(X, Y) DEF MOVIE　2，SPEED DEF MOVIE　3，VOLUME
DEF PARM	DEF PARM　実パラメータブロック名，［実パラメータ］…
DEF PEN	DEF PEN　{0［，ペンの太さ］｜1［，配列名］}
DEF SEG	DEF SEG［＝セグメントアドレス］
DEFSNG	DEFSNG　文字の範囲［，文字の範囲］
DEF SPRITE　形式1 形式2 形式3 形式4 形式5	DEF SPRITE　0，パターン番号，配列名，モード DEF SPRITE　1，キャラクタ番号，(sx, sy)，先頭パターン番号［,［横スプライト数］［,［縦スプライト数］［,［オフセット参照］［，色テーブル番号］]]] DEF SPRITE　2，色テーブル番号，配列名 DEF SPRITE　3［，色テーブル番号］ DEF SPRITE　99, 0
DEFSTR	DEFSTR　文字の範囲［，文字の範囲］
DEF USR	DEF USR　［数字］＝開始アドレス
DELETE　形式1 形式2	DELETE［行番号1］［{，｜－}［行番号2］］ DELETE［{行番号1｜ラベル名1}］［{，｜－}［{行番号2｜ラベル名2}］］
DELETES	DELETES　サブプログラム名
DIAL	DIAL　電話番号
DIM	DIM　変数名（添字の最大値［，添字の最大値］…）
DRAW	DRAW　グラフィック命令列
DSKF関数	DSKF（ドライブ番号）
DSKINI	DSKINI　ドライブ番号
DSKI$関数　形式1 形式2	DSKI$（ドライブ番号，トラック番号，セクタ番号［，媒体種別］） DSKI$（ドライブ番号，サイド，トラック番号，セクタ番号）
DSKO$関数　形式1 形式2	DSKO$　ドライブ番号，トラック番号，セクタ番号 DSKO$　ドライブ番号，トラック番号，セクタ番号，媒体種別

386	GEAR	V3.4	86HG	86	機　　　能
×	×	×	○	×	固定長文字列変数を定義し，固定長文字列領域を確保する．
○	○	×	×	×	変数の型をロング型整数に宣言する．
○	×	×	×	×	再生する左上座標を指定する． 再生速度を制限する． PCM音量を設定する．
×	×	×	○	×	一連の実パラメータを定義する．
○	×	×	×	×	線を描画するのに使用するペンの太さ，形状を設定する．
×	×	×	○	○	セグメントのベースアドレスを設定する．
○	○	○	○	○	変数の型を単精度実数に宣言する．
○	×	×	×	×	スプライトパターンを定義する． スプライトキャラクタを定義する． 色テーブルを定義する． パレット情報を色テーブルとして定義する． （SCREEN@ 0 のときのみ使用できる．） スプライトの定義情報をすべて消去する．
○	○	○	○	○	変数の型を文字に宣言する．
×	×	○	○	○	機械語プログラムの開始オフセットを指定する．
－ ○	× ×	○ ×	－ ○	－ ○	プログラムの行を削除する．
×	×	×	○	×	サブプログラムの解放を行う．
×	○	×	×	×	電話をかける．
○	○	○	○	○	配列変数の次元数と添字の最大値を指定し，その変数にメモリ領域を割り当てる．
×	×	×	○	○	ワールド座標内でグラフィックを描く．
○	×	○	○	○	ディスクの未使用領域のクラスタ数を返す．
×	×	○	×	×	フロッピィディスクのディレクトリの初期化を行う．
× ×	× ×	○ ×	× ○	× ○	指定されたセクタの内容をシステムランダムバッファに読み込む．
× ×	× ×	○ ○	○ ×	○ ×	システムランダムバッファの内容を，指定されたセクタに書き込む．

付録

命　　令	形　　式
形式 3	DSKO $ ドライブ番号，サイド，トラック番号，セクタ番号
DSP ON／OFF	DSP {ON｜OFF} 部品名
EDIT	EDIT 行番号 EDIT［開始行番号］［{,｜−} 終了行番号］ EDIT［開始ラベル名］［{,｜−} 終了ラベル名］
EFFECT	EFFECT 効果
ELS関数	ELS（レベル番号）
END	END
ENS$関数	ENS $（レベル番号）
ENV$	ENV $（環境変数名）
EOF関数	EOF（ファイル番号）
ERASE	ERASE 配列変数名［，配列変数名］…
ERL	ERL
ERR	ERR
ERROR	ERROR エラー番号
ERS関数	ERS（レベル番号）
EXEC	EXEC 開始アドレス
EXP関数	EXP（式）
FIELD	FIELD［#］ファイル番号，フィールド幅 AS 文字変数名［，フィールド幅 AS 文字変数名］…
FILES	FILES［"デバイス名"］［，L］ FILES［"ファイルディスクリプタ"］［，L］
FIND	FIND（文字式，部品名）
FIX関数	FIX（式）
FOR	FOR 制御変数＝初期値 TO 終値［STEP 増分値］
FRE関数　形式 1 形式 2 形式 3	FRE（{整数｜文字列}） FRE（{0｜1｜2｜3}） FRE（{0｜1｜2｜3｜4}）
GCURSOR	GCURSOR（x, y），（x2, y2）［,（x3, y3）…］［，色］

386	GEAR	V3.4	86HG	86	機　　能
×	×	×	○	○	
×	○	×	×	×	部品を画面に表示する／消す.
× × ×	× × ×	○ × ×	－ ○ ○	－ ○ ○	指定された行を画面に表示する.
×	○	×	×	×	効果（ZOOM,WIPEなど）の切り換え.
×	×	×	○	×	サブプログラムで発生したエラー行番号を返す.
○	○	○	○	○	プログラムの実行を終了する.
×	×	×	○	×	エラーの発生したサブプログラム名を返す.
×	×	×	○	○	環境変数に割り当てられている内容を取り出す.
○	×	○	○	○	ファイルの終わりを検出する.
○	×	○	○	○	配列変数をプログラムから消去する.
○	×	○	○	○	エラー発生時の行番号を返す
○	○	○	○	○	エラー発生時のエラー番号を返す.
○	×	○	○	○	BASICのエラー発生をシミュレートしたり，ユーザーのエラー番号の定義を可能にする.
×	×	×	○	×	サブプログラムで発生したエラー番号を返す.
×	×	○	○	○	機械語プログラムを実行する.
○	○	○	○	○	eを底とする指数関数の値を返す.
○	○	○	○	○	ランダムファイルのバッファに変数の領域を割り当てる.
－ ○	× ×	○ ×	－ ○	－ ○	指定されたデバイスのファイル名の一覧表（ディレクトリリスト）を出力する.
×	○	×	×	×	文字列を探し出す.
○	○	○	○	○	整数部分を返す.
○	○	○	○	○	NEXT命令までの一連の命令を繰り返し実行する.
× － ○	× － ○	○ ○ ×	× － ○	× － ○	メモリの未使用領域の大きさを返す.
×	×	○	○	○	画面上のグラフィックカーソルのドット座標を読み取る.

付録

命　　令		形　　　式
GET		GET［#］ファイル番号［，レコード番号］
GET@	形式 1 形式 2 形式 3 形式 4 形式 5 形式 6 形式 7	GET@（x1, y1）－（x2, y2），配列名 GET@（x1, y1）－（x2, y2），配列名［，色［，色］…］ GET@A（x1, y1）－（x2, y2），配列名 GET@A（x1, y1）－（x2, y2），配列名［，オフセット］ GET@（x1, y1）－（x2, y2），配列名，G［，色［，色］…］ GET@（x1, y1）－（x2, y2）配列名，{G|GC}［，色［，色］…］ GET@A（x1, y1）－（x2, y2），配列名，G GET@A（x1, y1）－{(x2, y2)|STEP（x, y)}，配列名，G
GETS$関数		GETS$（部品名，行番号）
GO		GO {ページ|ノート|部品名}
GOSUB （GOS.）	形式 1 形式 2	GOSUB　行番号 GOSUB {行番号|ラベル名}
GOTO （GO.）	形式 1 形式 2	GOTO　行番号 GOTO {行番号|ラベル名}
GUIDE		GUIDE（アイコン番号，質問内容，項目［，項目，項目…]）
HARDC		HARDC［機能］[,（X_1, Y_1）－（X_2, Y_2）]
HEX$関数		HEX$（式）
IF～THEN ～ELSE	形式 1 形式 2	IF　式　THEN{文|行番号}[ELSE {文|行番号}］または IF　式　GOTO　行番号 [ELSE {文|行番号}] IF　式　THEN {文［：文］…|行番号|ラベル名}[ELSE {文［：文］…|行番号|ラベル名}] または IF　式　GOTO {行番号|ラベル名}[ELSE {文［：文］…|行番号|ラベル名}]
IF～THEN ELSE ELSE IF THEN ENDIF		IF　式　THEN［コメント］ 文［：文］… ELSE IF　式　THEN［コメント］ 文［：文］… ELSE［コメント］ 文［：文］… ENDIF［コメント］

386	GEAR	V3.4	86HG	86	機　　　　能
○	○	○	○	○	ランダムファイルから指定されたレコードをバッファに読み込む.
－ ○ × ○ × × × ×	－ ○ × ○ × × × ×	○ × ○ × ○ × ○ ×	○ × ○ × － ○ ○ ○	○ × ○ × － ○ ○ 	画面上のキャラクタを配列に読み込む. グラフィック画面上のドットパターンを配列に読み込む. 画面上のキャラクタとその属性を配列に読み込む. グラフィック画面上のドットパターンを配列に読み込む. 画面上の指定色のドットパターンを配列に読み込む. 画面上の指定色のドットパターンを配列に読み込む. グラフィック画面上のドットパターンを配列に読み込む. グラフィック画面上のドットパターンを配列に読み込む.
×	○	×	×	×	テキストを読み込む.
×	○	×	×	×	分岐する.
－ ○	× ○＊	○ ×	－ ○	－ ○	サブルーチンを呼び出し，サブルーチン終了後はGOSUB命令の直後の命令に戻る.
－ ○	× ○＊	○ ×	－ ○	－ ○	無条件に指定された行番号へ分岐する.
×	○	×	×	×	ガイドウィンドウを表示，選択する.
○	×	○	○	○	画面データをプリンタに出力する.
○	○	○	○	○	整数を16進数の文字列に変換する.
－ ○	× ○＊	○ ×	－ ○	－ ○	式の値により，実行すべき文を選択する.
○	×	×	○	×	式の値により，実行すべき処理手続き（文の並び）を選択する.

＊：GEARでは行番号はない（ラベルのみ）

命　　令	形　　式
INKEY$	INKEY$
INP	INP（I／Oアドレス）
INPUT	INPUT［"プロンプトメッセージ"{,｜;}］変数名［，変数名］…
INPUT#	INPUT#ファイル番号，変数名［，変数名］…
INPUT$関数	INPUT$（文字数［,［#］ファイル番号］）
INSTR関数	INSTR（［検索開始位置,］文字式１，文字式２）
INT関数	INT（式）
INTERVAL	INTERVAL　割り込み間隔
INTERVAL ON／OFF／STOP	INTERVAL {ON｜OFF｜STOP}
JIS関数	JIS（文字式）
KACNV$関数	KACNV$（文字式）
KANJI ON／OFF	KANJI {ON｜OFF}
KEXT$関数	KEXT$（文字式，{0｜1}）
KEY	KEY　PEキー番号，文字列
KEY　LIST	KEY　LIST
KEY（n）ON／OFF／STOP	KEY（PFキー番号）{ON｜OFF｜STOP}
KILL	KILL　"ファイルディスクリプタ"
KINSTR関数	KINSTR（［検索開始位置,］文字列１，文字列２）
KLEFT$関数	KLEFT$（文字式，文字数）
KLEN関数	KLEN（文字式［,{0｜1｜2}］）
KMID$関数	KMID$（文字式，文字位置［，文字数］）
KNJ$関数	KNJ＄（式）
KOKUGO ON／OFF	KOKUGO {ON｜OFF}
KRIGHT$関数	KRIGHTS（文字式，文字数）

386	GEAR	V3.4	86HG	86	機　　　能
○	×	○	○	○	キーボードが押されていれば，その文字を返す．押されていなければ空文字を返す．
×	×	×	○	○	ハードウェアのI／Oレジスタからデータを読み取る．
○	×	○	○	○	キーボードから入力されるデータを読み取る．
○	×	○	○	○	ファイルからデータを読み込み，変数に代入する．
○	×	○	○	○	キーボードおよびファイルから指定された数の文字列を入力する．
○	○	○	○	○	文字列を探し出し，その位置をバイト数で返す．
○	○	○	○	○	式の値を越えない最大の整数を返す．
○	×	○	○	○	インターバルタイマ割り込みの間隔を指定する．
○	×	○	○	○	インターバルタイマ割り込みを許可，禁止，停止する．
○	○	○	○	○	文字列の最初の文字をJISコードに変換する．
○	○	○	○	○	文字列の中の日本語文字を，対応するANK文字に変換する．
×	×	○	○	○	漢字モードの設定や解除を行う．
○	○	○	○	○	文字列の中から日本語文字列またはANK文字列を取り出す．
○	×	○	○	○	キーに文字列を定義する．
○	×	○	○	○	キーに定義されている文字列を画面に表示する．
○	×	○	○	○	PFキーからの割り込みを許可，禁止，停止する．
○	×	○	○	○	ファイルを削除する．
○	○	○	○	○	文字列を探し出し，その位置を文字数で返す．
○	○	○	○	○	文字列の左から指定された文字数の文字列を取り出す．
○	○	○	○	○	文字列の文字数を返す．
○	○	○	○	○	文字列の任意の部分を取り出す．
○	○	○	○	○	式の値をJISコードとする1文字の日本語文字を返す．
×	×	○	○	○	国語モードの設定や解除を行う．
○	○	○	○	○	文字列の右から指定された文字数の文字列を取り出す．

付録

<table>
<tr><th>命　　令</th><th>形　　　　式</th></tr>
<tr><td>KTYPE関数</td><td>KTYPE（文字式，文字位置）</td></tr>
<tr><td>LEFT$関数</td><td>LEFT$（文字式，バイト数）</td></tr>
<tr><td>LEN関数</td><td>LEN（文字式）</td></tr>
<tr><td>LET</td><td>［LET］　変数名＝式</td></tr>
<tr><td>LINE　形式1
形式2
形式3
形式4</td><td>LINE［@］［(x1, y1)］－(x2, y2)，文字列［,［色］［,　{B|BF}］］
LINE［@］［(x1, y1)］－(x2, y2)，論理操作［,［色］　{［,［B］［，ラインスタイル］］|［，BF］}］
LINE［@］［{(wx1, wy1)|STEP（x1, y1)}］－{(wx2, wy2)|STEP（x2, y2)}，論理操作［,［パレットコード1］［,{［B］［，ラインスタイル］|BF［,{パレットコード2|タイルストリング}］］}］］
LINE［{(wx1, wy1)|STEP（x1, y1)}］－{(wx2, wy2)|STEP（x2, y2)}，論理操作［,［色］{［,［B］［，ラインスタイル］］|［, BF［,{中塗り色|タイルストリング}］］］}］</td></tr>
<tr><td>LINE INPUT</td><td>LINE　INPUT［"プロンプトメッセージ"{,|－}］文字変数</td></tr>
<tr><td>LINE INPUT#</td><td>LINE　INPUT#　ファイル番号，文字変数</td></tr>
<tr><td>LIST（L.）　形式1
形式2
形式3
形式4</td><td>LIST［行番号1］［{,|－}［行番号2］］
LIST［{行番号1|ラベル名1}］［{,|－}［{行番号2|ラベル名2}］］
LIST　"ファイルディスクリプタ"［,［行番号1］［{,|－}［行番号2］］
LIST　"ファイルディスクリプタ"［,［{行番号1|ラベル名1}］［{,|－}［{行番号2|ラベル名2}］］］</td></tr>
<tr><td>LLIST　形式1
形式2</td><td>LLIST［行番号1］［{,|－}［行番号2］］
LLIST［{行番号1|ラベル名1}］［{,|－}［{行番号2|ラベル名2}］］</td></tr>
<tr><td>LOAD（LO.）</td><td>LOAD　"ファイルディスクリプタ"［，R］</td></tr>
<tr><td>LOAD?</td><td>LOAD?［"［CASO：］ファイル名"］</td></tr>
<tr><td>LOAD@　形式1
形式2
形式3
形式4</td><td>LOAD@〔"ファイルディスクリプタ"〕
LOAD@　"ファイルディスクリプタ"　［,(x, y)］
LOAD@　"ファイルディスクリプタ"，配列名［,{0|1}］
LOAD@　"ファイルディスクリプタ"，配列名</td></tr>
<tr><td>LOADM</td><td>LOADM"ファイルディスクリプタ"［，オフセット値］［，R］</td></tr>
<tr><td>LOADS</td><td>LOADS　"ファイルディスクリプタ"</td></tr>
<tr><td>LOC関数</td><td>LOC（ファイル番号）</td></tr>
</table>

386	GEAR	V3.4	86HG	86	機　　　能
○	○	○	○	○	文字列中の文字のタイプを返す.
○	○	○	○	○	文字列の左から指定バイト数の文字を取り出す.
○	○	○	○	○	文字列の長さを返す.
○	○	○	○	○	右辺の式の結果を左辺の変数に代入する.
×	×	○	○	○	画面上にキャラクタを使って線および箱を表示する.
×	×	○	−	−	画面上にドットを使って線および箱を描く.
×	×	×	○	○	画面上にドットを使って線および箱を描く.
○	○	×	×	×	画面上にドットを使って線および箱を描く.
○	×	○	○	○	1行全体の文字列（255文字以内）を区切ることなく文字変数に読み込む.
○	×	○	○	○	指定されたファイルから1行を区切ることなく読み込み，変数に代入する.
−	×	○	−	−	メモリにあるプログラムの全部または一部をCRTに出力する.
○	×	×	○	○	同上
−	×	○	−	−	メモリにあるプログラムの全部または一部を指定したファイルに出力する.
○	×	×	○	○	同上
− ○	× ×	○ ×	− ○	− ○	メモリにあるプログラムの全部または一部をプリンタに出力する.
○	×	○	○	○	プログラムファイルをメモリに読み込む.
×	×	○	×	×	カセットテープの内容とチェックサムの照合を行う.
○	○	×	×	×	音色データファイル.PMB／.FMBを読み込む.
○	○	×	×	×	イメージファイル．TIF／.JPGを読み込む
○	×	×	×	×	EUPファイルを読み込む
○	×	×	×	×	配列に.EUP以外のデータを読み込む.
○*	○*	○	○	○	機械語プログラムファイルをメモリに読み込む. *386およびGEARではRの指定はない.
×	×	×	○	×	サブプログラムをメモリ上に読み込む.
○	○	○	○	○	ランダムファイルの，次にGETまたはPUTされるレコード番号を返す.

命　　令	形　　式
LOCATE　形式1 　　形式2	LOCATE[水平位置][,[垂直位置][,カーソル表示スイッチ]] LOCATE[水平位置][,[垂直位置][,[カーソル表示スイッチ][,カーソルエンド]]]
LOF関数	LOF（ファイル番号）
LOG関数	LOG（式）
LPOS関数	LPOS（式）
LPRINT	LPRINT［式［{,｜;}［式]］…］
LPRINT USING	LPRINT　USING　書式文字列；［式［{,｜;}［式]］…］
LSET／RSET	{LSET｜RSET} 文字変数＝文字式
MAP関数	MAP（座標値，機能）
MERGE （ME.）	MERGE　"ファイルディスクリプタ"［，R］
MID$関数	MID$（文字式，文字位置［，バイト数］）
MKD$関数	MKD$（倍精度実数）
MKDMBF$関数	MKDMBF$（倍精度実数）
MKI$関数	MKI$（整数）
MKL$関数	MKL$（ロング型整数）
MKS$関数	MKS$（単精度実数）
MKSMBF$関数	MKSMBF$（単精度実数）
MODE$	MODE$（運用環境変数）
MON	MON
MOTOR（M.）	MOTOR［{ON｜OFF}］
MOUSE　形式1 　　形式2 　　形式3 　　形式4 　　形式5 　　形式6 　　形式7	MOUSE 0［，マウス表示ページ］ MOUSE 1［,[sx]［,[sy]［,[カーソルスイッチ]]］］ MOUSE 2，andパターン，ドットパターン[,水平読み取り位置，垂直読み取り位置］ MOUSE 3，移動方向，移動比率 MOUSE 4，sx1，sy1，sx2，sy2 MOUSE 5 MOUSE 6，モード，andパターン，ドットパターン［，水平読み取り位置，垂直読み取り位置］

386	GEAR	V3.4	86HG	86	機　　　　能
○ ×	× ×	○ ×	− ○	− ○	画面上の任意の位置にカーソルを移動する.
○	○	○	○	○	入力バッファ中の文字数を返すか，または，ランダムファイルの最大レコード番号を返す.
○	○	○	○	○	自然対数の値を返す.
○	×	○	○	○	プリンタバッファ内の水平文字位置を返す.
○	×	○	○	○	文字または数値をプリンタへ出力する.
○	×	○	○	○	指定した書式で文字または数値をプリンタに出力する.
○	○	○	○	○	文字データーをランダムファイルのバッファに代入する.
○	×	×	○	○	スクリーン座標とワールド座標間の相互変換を行う.
○	×	○	○	○	メモリにあるプログラムと，指定されたファイルのプログラムを混ぜ合わせる.
○	○	○	○	○	指定バイト長の文字列を取り出す.
○	○	○	○	○	倍精度型の数値を文字列に変換する.
○	×	×	×	×	倍精度型の数値をマイクロソフト内部形式に変換し，8バイトの文字列として扱う.
○	○	○	○	○	整数型の数値を文字列に変換する.
○	○	×	×	×	ロング型整数型の数値を文字列に変換する.
○	○	○	○	○	単精度型の数値を文字に変換する.
○	×	×	×	×	単精度型の数値をマイクロソフト内部形式に変換し，4バイトの文字列として扱う.
×	×	×	○	×	運用環境変数名で指定された運用環境設定値を取り出す.
×	×	○	×	×	BASICコマンドモードからモニタに移る.
×	×	○	×	×	カセットテープレコーダのモータ制御をする.
○ ○ ○ ○ ○ ○ ○	○ ○ ○ ○ ○ × ×	× × × × × × ×	○ ○ ○ ○ ○ ○ ×	○ ○ ○ ○ ○ ○ ×	マウス機能を初期化する. マウスカーソルの位置の設定や，表示／消去の選択を行う. マウスカーソルの形状や，カーソル内の読み取り位置を設定する. マウスカーソルの移動比率を設定する. マウスカーソルの移動範囲を設定する. マウスの使用を終了する. マウスカーソルの形状を単色またはカラー32×32ドットに設定する.

命　　令	形　　　　式
MOUSE関数　形式 1 形式 2	MOUSE（式） MOUSE（式，ボタン番号）
MOUSE（n）ON／OFF／STOP	MOUSE（条件番号）{ON|OFF|STOP}
MOVIE CLOSE	MOVIE CLOSE
MOVIE INFO　形式 1 形式 2	MOVIE INFO 1，ロング型整数配列名 MOVIE INFO 2，文字列配列名
MOVIE OPEN	MOVIE OPEN　"ファイル名．拡張子"
MOVIE PLAY	MOVIE PLAY [，開始フレーム [，終了フレーム]]
NAME	NAME　"旧ファイルディスクリプタ" AS　"新ファイル名"
NEW	NEW
NEXT	NEXT [制御変数 [，制御変数] …]
OCT$関数	OCT $（式）
ON COM（n）GOSUB	ON COM（ポート番号）　GOSUB {行番号 | ラベル名}
ON ERROR GOTO	ON ERROR GOTO {行番号 | ラベル名}
ON～GOSUB	ON　式　GOSUB {行番号 | ラベル名} [，{行番号 | ラベル名}] …
ON～GOTO	ON　式　GOTO {行番号 | ラベル名} [，{行番号 | ラベル名}] …
ON INTERVAL GOSUB	ON INTERVAL GOSUB {行番号 | ラベル名}
ON KEY（n）GOSUB	ON KEY（PFキー番号）GOSUB {行番号 | ラベル名}
ON MOUSE（n）GOSUB	ON MOUSE（条件番号）GOSUB {行番号 | ラベル名}
ON TIME GOSUB	ON TIME GOSUB {行番号 | ラベル名}
OPEN　形式 1 形式 2	OPEN　"モード"，[#] ファイル番号，"ファイルディスクリプタ" OPEN　"ファイルディスクリプタ" [OPENモード] [ALLOWモード] AS [#] ファイル番号
OUT	OUT　I／O　アドレス，式

386	GEAR	V3.4	86HG	86	機　　能
○ ○	○ ○	× ×	○ ○	○ ○	マウスカーソルの情報を返す. マウスのボタンの状態またはマウスカーソルの情報を返す.
○	×	×	○	○	マウスによる割り込みを許可，禁止，停止する.
○	×	×	×	×	OPENしたムービーファイルをCLOSEする.
○ ○	○ ×	× ×	× ×	× ×	OPENしたムービーファイルの情報を得る.
○	×	×	×	×	ムービーファイルをOPENする.
○	○	×	×	×	ムービーファイルを再生する.
○	×	○	○	○	ファイル名を変更する.
○	×	○	○	○	メモリにあるプログラムを消去し，すべての変数を初期化する.
○	○	○	○	○	FOR～NEXTループの終わりを指定する.
○	○	○	○	○	整数を8進数の文字列に変換する.
○	×	○*	○	○	通信回線からの入力割り込み時の処理ルーチンを定義する. *V3.4ではラベル名はない.
○	○*	○*	○	○	エラートラップ機能を可能にする. *GEARでは行番号はない. V3.4ではラベル名はない.
○	○*	○*	○	○	式の値により，指定された行番号をもつサブルーチンを呼び出す. *GEARでは行番号はない. V3.4ではラベル名はない.
○	○*	○*	○	○	式の値に指定された行番号へ分岐する. *GEARでは行番号はない. V3.4ではラベル名はない.
○	×	○*	○	○	インターバルタイマ割り込み処理ルーチンを定義する. *V3.4ではラベル名はない.
○	×	○*	○	○	PFキー割り込み処理ルーチンを定義する. *V3.4ではラベル名はない.
○	×	×	○	○	マウス割り込み処理ルーチンを定義する.
○	×	○	○	○	タイマ割り込み処理ルーチンを定義する.
○ ○	△* ×	○ ×	○ ○	○ ○	ファイルのオープン処理を行う. *GEARではランダムアクセスファイルのみサポート
○	×	×	○	○	ハードウェアのI／Oレジスタにデータを書き込む.

付録

命　　令	形　　式
OUTM	OUTM［#ポート番号，］式［，式］…
PAD関数	PAD（パッドポート番号）
PAINT　形式1 形式2 形式3 形式4 形式5 形式6	PAINT（x,y）［,［色］［，境界色1［，境界色2］…]］ PAINT（x,y），タイルストリング［，境界色1［，境界色2］…］ PAINT［@］{(x,y) | STEP（x,y)}［,［色］［，境界色1［，境界色2］…］ PAINT［@］{(x,y) | STEP（x,y)}，タイルストリング［，境界色1［，境界色2］…］ PAINT {(x,y) | STEP（x,y)}［,{色 | タイルストリング}］［，境界色1］［，境界色2］… PAINT@ {(x,y) | STEP（x,y)}［,{色 | タイルストリング}］
PALETTE	PALETTE［パレット番号，{色コード | [G，R，B]}］
PALETTE@	PALETTE@［パレット番号1］［,［パレット番号2］］
PARM	PARM　仮パラメータ［，仮パラメータ］…
PART	PART　パート，チャンネル
PART関数	PART（パート）
PASTEL	PASTEL［比率］
PCMPLAY	PCMPLAY　配列名［，音量］
PCMREC	PCMREC　配列名，サンプリング周波数
PEEK関数　形式1 形式2	PEEK（式） PEEK（メモリアドレス［，型番号）
PLAY	PLAY［#音源,］"MML"［，"MML"］
PLAY関数	PLAY（0） PLAY（1） PLAY（2）
PLAY@	PLAY@配列名［,{0 | 1}］
PLAY ON／OFF／STOP	PLAY {ON|OFF|STOP}
POINT	POINT {(wx, wy) | STEP（x,y)}
POINT関数　形式1 POINT関数　形式2	POINT（x,y） POINT（機能）

386	GEAR	V3.4	86HG	86	機　　　能
○	×	○	×	×	MIDIポートへデータを送る．
○	○	△	×	×	パッドの方向を返す．V3.4ではSTRIG
− − × × ○ ○	− − × × ○ ○	○ ○ × × × ×	− − ○ ○ × ×	− − ○ ○ × ×	指定された境界で囲まれた範囲を指定された色または タイルパターンで塗る． 連続している同じ色の範囲を塗り直す．
○	○	○	○	×	パレットの色を設定する．
×	×	○	×	×	パレットを初期状態（電源投入直後）にする．
×	×	×	○	×	サブプログラムの呼び出し元によって指定された実パラメータを受け取る（パラメータの定義）．
○	○	×	×	×	PLAY命令で指定するパートをチャンネルに割り当てる．
○	○	×	×	×	PLAY命令で指定するパートが割り当てられているチャンネル番号を返す．
○	×	×	×	×	描画色の表示色との混合比率を設定する．
○	×	×	×	×	PCM音声データをPCM音源に送って発声させる．
○	×	×	×	×	マイクから取り込んだ音声をサンプリングして，データを配列に入れる．
× ○	× ○	○ ×	○ ×	○ ×	メモリの内容を読み出す．
○	○	○	×	×	音楽の演奏を行う．386には［#音源，］はない．
○ ○ ○	○ ○ ×	○ × ×	× × ×	× × ×	PLAY文の演奏中かどうかの状態を返す． 演奏中の小節番号を返す． PCMPLAYで再生中かどうかを返す．
○	×	×	×	×	EUPファイルを演奏する．
○	○	○	×	×	PLAYの演奏を許可，中止（初期化），一時停止する．
○	×	×	○	○	最終参照座標を変更する．
○ ○	× ×	○ ×	○ ○	○ ○	指定した位置のドットの状態を返す． 最終参照座標の値を返す．

命　令		形　式
POKE	形式1 形式2	POKE　書き込みアドレス（オフセット），データ POKE　メモリアドレス，書き込む値［，型番号］
POS関数		POS（ファイル番号）
PRESET		PRESET（x, y） PRESET {(wx, wy) | STEP（wy）}
PRINT（?）		PRINT［式［{, | ; }［式］］…］
PRINT#	形式1 形式2	PRINT#ファイル番号［，出力ならび］ PRINT#ファイル番号，USING書式文字列；［出力ならび］
PRINT@		PRINT@［(x, y),］漢字コード［{, | ; }［漢字コード］］…
PRINT USING		PRINT USING書式文字列；［出力ならび］
PSET		PSET（x, y［,［色］［，論理操作］］） PSET {(wx, wy | STEP（x, y)}［,［色］［，論理操作］］
PTRIG関数		PTRIG（パッドポート番号）
PUT		PUT［#］ファイル番号［，レコード番号］
PUT@	形式1 形式2 形式3 形式4 形式5 形式6 形式7	PUT@（x1, y1）−（x2, y2），配列名［，色］ PUT@A（x1, y1）−（x2, y2），配列名 PUT@（x1, y1）−（x2, y2），配列名［，論理操作［，色］］ PUT@（x1, y1）−{(x2, y2) | STEP（x, y)}，配列名，論理操作［，色］ PUT@A（x1, y1）−（x2, y2），配列名，論理操作 PUT@A（x1, y1）−{(x2, y2) | STEP（x, y)}，配列名，論理操作 PUT@A（x1, y1）−（x2, y2），配列名［,［論理操作］［,［横倍率］［，縦倍率］［，透過色］［，オフセット］］］］
PUTS		PUTS（部品名，文字式，行番号）
RANDOMIZE（RNDM.）		RANDOMIZE［式］
READ		READ　変数名［，変数名］…
REM（'）		REM　注釈文字列
RENUM	形式1	RENUM［新行番号］［,［旧行番号］［，増分値］］

386	GEAR	V3.4	86HG	86	機　　　能
× ○	× ○	○ ×	○ ×	○ ×	メモリの指定番地にデータを書き込む.
○	×	○	○	○	カーソルまたはプリンタバッファの水平位置を返す.
− ○	− ○	○ ×	− ○	− ○	画面上の任意の位置のドットに背景色を設定する.
○	×	○	○	○	画面に式の結果を出力する.
○ ○	× ×	○ ○	○ ○	○ ○	指定されたファイルに式の結果を出力する.
×	×	○	○	○	漢字を画面に表示する.
○	×	○	○	○	指定した書式で文字または数値を画面に出力する.
− ○	− ○	○ ×	− ○	− ○	画面上の任意の位置にドットを設定する.
○	○	△	×	×	パッドのボタンが押されているかどうかを返す. V3.4ではSTICK
○	○	○	○	○	指定されたランダムファイルにバッファの内容を書き込む.
×	×	○	○	○	GET@で読み込まれたキャラクタを任意の位置に表示する.
×	×	○	○	○	同上
○	○	○	−	−	GET@で読み込まれたドットパターンを任意の位置に表示する.
×	×	×	○	○	同上
−	×	○	−	−	GET@Aで読み込まれたドットパターンを任意の位置に表示する.
−	×	×	○	○	同上
○	○	×	×	×	同上
×	○	×	×	×	文字列を書き込む.
○	○	○	○	○	乱数の系列を変更する.
○	○	○	○	○	DATA命令で定義した定数を変数に読み込む.
○	○	○	○	○	プログラム中の注釈.
△	×	○	−	−	プログラムの各行の行番号を付け直す.

命　　令	形　　式
形式 2	RENUM [新行番号] [,[{旧行番号 | ラベル名}] [, 増分値]]
REPLACE	REPLACE 変更値 TO 置換値 [, 部品名]
RESET	RESET
RESTORE 形式 1 形式 2	RESTORE [行番号] RESTORE [{行番号 | ラベル名}]
RESUME	RESUME [{NEXT | 行番号 | ラベル名}]
RETURN (RET.)	RETURN [{行番号 | ラベル名}]
RETURNS	RETURNS
RIGHT$関数	RIGHT$ (文字式, バイト数)
RND関数	RND [(式)]
ROLL	ROLL [上方向ドット数] [,[左方向ドット数] [, オプション]]
RUN 形式 1 形式 2 形式 3	RUN [行番号] RUN [{行番号 | ラベル名}] RUN"ファイルディスクリプタ" [, R]
SAVE	SAVE "ファイルディスクリプタ" [, A] SAVE "ファイルディスクリプタ" [, {A|P}]
SAVE@ 形式 1 形式 2 形式 3	SAVE@ "ファイルディスクリプタ" SAVE@ " ファイルディスクリプタ", (開始位置) − (終了位置) [, [パレット情報] [, 圧縮情報 [, [YCbCr] [, [Y] [, C]]]]] SAVE@ "ファイルディスクリプタ", 配列名
SAVEM	SAVEM "ファイルディスクリプタ", 開始アドレス, 終了アドレス, 入口アドレス
SCREEN関数	SCREEN (x, y [, セレクトスイッチ])
SCREEN 形式 1 形式 2 形式 3	SCREEN { 0 | 1 [,[アクティブページ] [,[ディスプレイページ] [,プライオリティ]]]} SCREEN [画面モード] [,[アクティブページ] [,[ディスプレイページ] [,,, ビデオ出力コード]]] SCREEN [アクティブVRAMコード] [,[ディスプレイVRAMコード] [,[アクティブページ] [,[ディスプレイページ]]]]
SCREEN@	SCREEN@ 画面モード

386	GEAR	V3.4	86HG	86	機　　　　　能
○	×	×	○	○	行番号を付け直す.
×	○	×	×	×	文字列を置き換える.
×	×	×	○	○	ファイル番号とディスクファイルの切り離しを行う.
－ ○	× ○＊	○ ×	－ ○	－ ○	READ命令の読み込み対象となるDATA命令を指示する. ＊GEARでは行番号がない.
○	○＊	○＊	○	○	エラー処理を終了して，プログラムの実行を開始する. ＊V3.4ではラベル名がない. GEARでは行番号がない.
○	○＊	○＊	○	○	サブルーチンを終了し，元のプログラムへ復帰する. ＊V3.4ではラベル名がない.
×	×	×	○	×	サブプログラムからの復帰を行う.
○	○	○	○	○	文字列の右から指定バイト数の文字を取り出す.
○	○	○	○	○	乱数を返す.
○	×	×	○	○	VRAMの内容を上下，左右へスクロールする.
－ ○ ○	× × ×	○ × ○	－ ○ ○	－ ○ ○	メモリまたはファイルにあるプログラムを実行する.
○ ×	× ×	－ ○	－ ○	－ ○	メモリ内にあるプログラムをファイルに保存する.
○ ○ ○	○ ○ ○	× × ×	× × ×	× × ×	音色データをファイル.FMBに保存する. イメージデータをファイル.TIF／.JPGに保存する. 配列のデータをファイルに保存する.
×	×	○	○	○	メモリ内の機械語プログラムをファイルに保存する.
○	×	○	○	○	指定座標のキャラクタコードおよび属性を返す.
○ × ×	× × ×	× × ○	× ○ ×	× ○ ×	グラフィック2画面モードに切り替える. グラフィック画面の画面モードやページの設定をする. グラフィック画面のページの設定をする.
○	×	○	×	×	画面モードを設定する.

付録

命　　令	形　　式
SEARCH	SEARCH（配列変数名，検索データ［,［開始要素番号］［，ステップ数］］）
SETENV	SETENV　環境定義文字列
SETMODE	SETMODE　運用環境変更文字列
SGN関数	SGN（式）
SHELL　形式1 形式2	SHELL "ファイルディスクリプタ" ［,［パラメータ列］［，RF］ SHELL ｛文字定数｜文字変数｝
SIMPOSE　形式1 形式2	SIMPOSE ｛ON ［輝度］｜OFF｝ SIMPOSE ［モード］［,［輝度］］
SIN関数	SIN（式）
SINPUT	SINPUT
SKIPF	SKIPF ［"［CASO：］ファイル名" ］
SMSGPLAY	SMSGPLAY　サウンドメッセージID
SOUND	SOUND　OPNのレジスタ番号，データ
SPACE$関数	SPACE$（個数）
SPC関数	SPC（個数）
SPRITE　形式1 形式2 形式3 形式4 形式5 形式6 形式7	SPRITE 0，キャラクタ番号，表示 SPRITE 1，キャラクタ番号，パターン番号 SPRITE 2，キャラクタ番号，色テーブル番号 SPRITE 3，キャラクタ番号，角度 SPRITE 4，キャラクタ番号，縮小 SPRITE 5，キャラクタ番号，スーパーインポーズ SPRITE 6，［,［キャラクタ番号］［,［x］［，y］］］
SPRITE関数	SPRITE　（キャラクタ番号，番号）
SPRITE ON／OFF	SPRITE ｛ON ［色］｜OFF｝

386	GEAR	V3.4	86HG	86	機　　　　能
○	×	○	○	○	配列変数の要素を探し出し，要素番号を返す．
×	×	×	○	○	チャイルドプロセスに渡す環境を設定する．
×	×	×	○	×	BASICの運用環境の一部を変更する．
○	○	○	○	○	正負を判定する．
× ○	× ×	× ×	○ ×	○ ×	チャイルドプロセス上でプログラムを読み込み，実行する．
○ ○	× ×	× ○	× ×	× ×	コンピュータ画像とテレビ画像の重ね合わせ（スーパーインポーズ）を制御する．
○	○	○	○	○	サインの値を返す．
○	×	○	×	×	テレビ，またはビデオの画像をVRAMに読み込み，表示する．
×	×	○	×	×	指定したファイルの次にカセットテープを進める．
○	×	×	×	×	サウンドメッセージを鳴らす．
×	×	○	×	×	PSG（Programmable Sound Generator）を直接，コントロールする．
○	○	○	○	○	空白を返す．
○	×	○	○	○	指定された数の空白を出力する．
○ ○ ○ ○ ○ ○ ○	× × × × × × ×	× × × × × × ×	× × × × × × ×	× × × × × × ×	スプライトキャラクタを表示／消去する． 指定したスプライトキャラクタのパターンを変更する． 指定したスプライトキャラクタに割り当てられた色テーブル番号を変更する． 指定したスプライトキャラクタを回転または左右反転させる． 指定したスプライトキャラクタを縮小する． 指定したスプライトキャラクタをスーパーインポーズ表示する． 指定したスプライトキャラクタを移動させる．
○	×	×	×	×	指定したスプライトキャラクタの情報を返す． 0：表示／消去　　6：表示位置のX座標 1：先頭パターン番号　　7：表示位置のY座標 2：色テーブル番号　　8：横スプライト数 3：回転および左右反転　　9：縦スプライト数 4：縮小 5：スーパーインポーズ表示
○	×	×	×	×	テキスト画面とスプライト画面の切り替えを行う．

命　　令	形　　　式
SPRITE SCREEN　形式1 形式2	SPRITE SCREEN (x, y) SPRITE SCREEN　{0|1}
SPRITE TIME	SPRITE TIME
SQR関数	SQR（式）
STOP	STOP
STOP ON／OFF	STOP {ON|OFF}
STRING$関数	STRING$（文字数，{キャラクタコード|文字式}）
STR$関数	STR $（式）
SUBPGM	SUBPGM
SWAP	SWAP　変数，変数
SYMBOL　形式1 形式2 形式3	SYMBOL (x, y)，文字列，横倍率，縦倍率 [, [色] [, [角度コード] [, 論理操作]]] SYMBOL {(wx, wy)|STEP (x, y)}，文字列，横倍率，縦倍率 [, [色] [, [角度コード] [, 論理操作]]] SYMBOL {(wx, wy)|STEP (x, y)}，文字列，横倍率，縦倍率 [, [色] [, [角度コード] [, 論理操作] [, フォント] [, 文字間隔]]]]
SYMBOL@	SYMBOL@ (x, y)，日本語文字列，横倍率，縦倍率，[, [色] [, [角度コード] [, [論理操作]]]] SYMBOL@ {(x, y)|STEP (x, y)}，日本語文字列，横倍率，縦倍率，[, [色] [, [角度コード] [, [論理操作]]]]
SYSTEM	SYSTEM
TAB関数	TAB（桁位置）
TALK	TALK "VML"
TAN関数	TAN（式）
TERM	TERM ["cbpsmalx"]
TIME	TIME"hh:mm:ss"
TIME関数	TIME
TIME ON／OFF／STOP	TIME {ON|OFF|STOP}

386	GEAR	V3.4	86HG	86	機　　　　　　　能
○	×	×	×	×	グラフィック画面上のどの位置にスプライト画面を表示するかを指定する.
○	×	×	×	×	32768色モードにおけるスプライト画面の大きさを変える.
○	×	×	×	×	スプライト表示のタイミングをとる.
○	○	○	○	○	平方根を返す.
○	○	○	○	○	プログラムの実行を中断する.
○	○	○	○	○	BREAKキーによるプログラムの中断を無効または有効にする.
○	○	○	○	○	指定された文字の文字列を返す.
○	○	○	○	○	数値を表す文字列を返す.
×	×	×	○	×	サブプログラムの宣言を行う.
○	×	○	○	○	2つの変数の値を交換する.
−	×	○	−	−	画面上の任意の位置に文字列を指定角度，指定サイズで表示する.
−	×	×	○	○	
○	×	×	×	×	注）文字列は日本語も可.
×	×	○	−	−	画面上の任意の位置に文字列を指定角度，指定サイズで表示する.
×	×	×	○	○	
○	○	×	○	○	BASICを終了し，システムに制御を戻す.
○	×	○	○	○	指定された桁位置まで空白を出力する.
×	×	○	○	○	音声を出力する. （VML:Voice Macro Language 音声合成マクロ言語）
○	○	○	○	○	タンジェントの値を返す.
×	×	○	△	△	動作モードをターミナルモードにする.
○	○	○	○	○	タイマ割り込みの時刻を設定する.
○	×	○	○	○	システムタイムの時刻を秒で返す.
○	×	○	○	○	タイマ割り込みを許可，禁止，停止する.

命　　令	形　　式
TIME$	TIME$［="hh：mm：ss"］
TROFF	TROFF
TRON	TRON
UNLIST	UNLIST　行番号
UPDATE	UPDATE［#］ファイル番号
USR	USR［数字］（引数）
VAL関数	VAL（文字式）
VARPTR　形式1 形式2	VARPTR（変数名） VARPTR{(SYSTEM}［，機能］
VERSION$	VERSION$
VIEW	VIEW［(x1, y1)－(x2, y2)［，領域色］［，境界色］］］
VIEW	VIEW　部品名
VIEW END	VIEW END
VIEW関数	VIEW（機能）
VOICE　形式1 形式2	VOICE　整数配列名1［，整数配列名2［，整数配列名3］］ VOICE　音色番号，配列名［，音源］
VOICE COPY	VOICE COPY　音色番号，整数配列名 VOICE COPY　音色番号，配列名［，音源］
VOICE LFO	VOICE LFO　チャネル番号［，ウエーブフォーム［，シンクロ［，スピード［，PMO［，AMO［，PMS［，AMS［，ディレイタイム］］］］］］］
VOICE SET	VOICE SET　配列名
WAIT	WAIT（時間）
WEND	WEND
WHILE	WHILE 式
WIDTH①	WIDTH［画面の桁数］［，画面の行数］
WIDTH②	WIDTH｛#ファイル番号｜"デバイス名"｝，サイズ
WINDOW	WINDOW［{(wx1, wy1)｜STEP（x1, y1)}］－［{(wx2, wy2)｜STEP（x2, y2)}］
WINDOW関数	WINDOW（機能）
WRITE	WRITE［式［{，｜；}［式］］…］
WRITE#	WRITE#ファイル番号［［，式［{，｜；}［式］］…］

386	GEAR	V3.4	86HG	86	機　　　　　　　　能
○	○	○	○	○	内蔵タイマの示す時刻を返すかまたは変更する.
○	×	○	○	○	プログラムの実行状態の追跡を止める.
○	×	○	○	○	プログラムの実行状態を追跡する.
×	×	○	○	○	非表示番号を指定する.
×	×	×	○	×	ランダムファイルのレコードを更新する.
×	×	○	○	○	引数よりユーザのアセンブリ言語ルーチンを呼び出す.
○	○	○	○	○	数字の文字列を数値に変換する.
○ ×	○ ×	○ ×	○ ○	○ ○	変数のアドレスを返す.
×	○	×	×	×	TownsGEARのバージョン番号を返す.
○	×	×	○	○	ウィンドウの表示範囲（ビューポート）を指定する.
×	○	×	×	×	グラフィック処理の開始を宣言する.
×	○	×	×	×	グラフィック処理の終了を宣言する.
○	×	×	○	○	現在，ビューポートが設定されている位置を返す.
× ○	× ○	○ ×	× ×	× ×	音色を設定する.
– ○	– ○	○ ×	× ×	× ×	音色データを配列変数にコピーする.
×	×	○	×	×	チャネルに対してLFO効果を与える.
○	○	×	×	×	配列のPCM音声データをPLAY文で演奏するための音色データとして設定する.
○	○	×	×	×	指定時間だけプログラムを中断する.
○	○	○	○	○	WHILE～WENDループの終わりを示す.
○	○	○	○	○	一連の命令を条件つきで繰り返し実行する.
○	×	○	○	○	画面上に表示する文字の行数と桁数を指定する.
○	×	○	○	○	デバイスの1行の長さを指定する.
○	×	×	○	○	ワールド座標内のウィンドウの範囲を指定する.
○	×	×	○	○	現在，ウィンドウが設定されている位置を返す.
○	×	○	○	○	画面に式の結果を出力する.
○	×	○	○	○	式の結果を指定されたファイルに出力する.

付録8　命令の追加・変更について

F－BASIC 386のレベルアップで追加変更された命令は以下のとおりです。追加変更された命令は、それ以前のレベルのBASICでは使用できません。

1．V1.1L21の命令追加変更

①　DEF KANJI(追加)

②　INP関数(追加)

③　OUT(追加)

④　PEEK；POKE(変更)

L20以前では、セレクタを(セレクタ)のように括弧付きで指定しますが、L21以降では括弧無しで指定します。L20以前のプログラムをL21以降で実行する場合は、修正してから実行してください。

2．V2.1L10の命令追加変更

①　DEF FONT(追加)

②　IF～THEN；ELSE；ELSE IF～THEN(追加)

③　LOAD@(変更)

・EUPファイルの読み込みを可能にする(形式3)

・画像圧縮したデータの読み込みを可能にする(形式2)

・JPEGファイルの読み込みを可能にする。

④　OPEN(変更)

・ALLOWモードの指定を追加

⑤　PLAY@(追加)

⑥　SAVE@(変更)

・画像データを圧縮して格納できる。

・JPEGファイルに格納できる。

⑦　SCREEN(追加)

⑧　MOVIE(追加)

⑨　SMSGPLAY(追加)

⑩　CLEAR(変更)

・DLL領域の大きさを指定する。

⑪　MOUSE 0(変更)

・マウスカーソルを描画するページを指定する。

⑫　MOUSE 6(追加)

⑬　HARDC(変更)

・TownsOSのデバイスドライバを使用します。設定の中のハードコピーを実行してからBASICを起動してください。

●詳しくは各命令の説明を参照してください。

付録

五十音順索引

あ

い

う

え

お

か

き

く

こ

さ

し

す

せ

た

ち

つ

て

と

に

は

ひ

ふ

へ

ほ

ま

む

め

も

よ

ら

り

ろ

わ

アルファベット順索引

A

C

D

E

F

G

I

J

K

L

その他

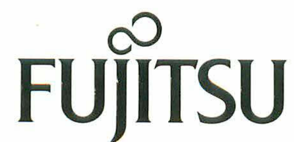
FMTOWNS
F-BASIC386 V2.1リファレンス
81SP-0442-1-0

発 行 日　1992年11月

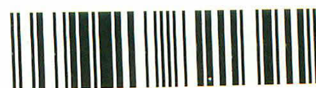
発行責任　富士通株式会社

Printed in Japan

Ⓐ 9311-10



本マニュアルは、100%リサイクル可能な用紙を使用しています。



81SP-0442-1